ÍNDICE

Comandos da Linguagem Clipper 5.2

?/??

Propósito: Mostrar um ou mais valores na console (vídeo) ou impressora.

Sintaxe: ?/?? <Lista de expressões>

Exemplo:

```
CLEAR             / /  limpa a tela
? “Exemplo do comando ?”       / / exibe a informação no vídeo
? date ( )                 / /  exibe a data ( nova linha )
? ”a data de hoje é..:”
?? date ( )     / / exibe na mesma posição anterior do cursor
```

@... BOX

Propósito: Construir um box (caixa) na tela.

Sintaxe: @ <Lin inicial>,<Col inicial>, <Lin final>, <Col final>, BOX <Cadeia>

Exemplo:

LOCAL C,L

```
/*    MODULO :    M.PRG
      FUNÇÃO:     ACESSAR TODOS OS PROGRAMAS
*/
SAVE SCREEN TO TECLADOS
CLEAR
SET DATE BRIT
SET CONFIRM ON
SET DELETE ON
      DO WHILE .T.
CLEAR
SET WRAP ON
SET MESSAGE TO 23 CENTER
SET COLOR TO B/W
REBOX=CHR (201) +CHR (205) +CHR (187) +CHR (186) +;
       CHR (188) +CHR (205) +CHR (200) +CHR (186)
PRIVATE=EMPRESA:= “FACULDADES REUNIDAS LTDA”
L = 08
C = 22
CLEAR
```

```
@ 00,00,03,39 BOX RETBOX
@ 00,40,03,79 BOX RETBOX
@ 04,00,21,79 BOX RETBOX+CHR (177)
@ 22,00,24,79 BOX RETBOX
@ 01,02 SAY EMPRESA
@ 01,42 SAY "CONTROLE DE FACULDADE"
@ 01,70 SAY DATE( )
@ 02,42 SAY  "MODULO PRINCIPAL"
@ 02,70 SAY TIME ( )
@ 23,02 SAY "MENSAGEM"
@ L-1,C-2,L+7,C+35 BOX RETBOX+CHR(255)
SET COLOR TO
@ L,C       PROMPT "PROCESSAR FACULDADES"
@ L+2,C     PROMPT "PROCESSAR TABELA DE CURSOS"
@ L+4,C     PROMPT "PROCESSAR ALUNO"
@ L+6,C     PROMPT "VOLTAR AO D.O.S."
MENU TO OPC
DO CASE
      CASE OPC = 1
            DO MENUFACU
      CASE OPC = 2
            DO MENUCURS
      CASE OPC = 3
            DO MENUALUN
      OTHERWISE
            RESTORE SCREEN FROM TELA TECLADOS
            CANCEL
ENDCASE
ENDDO
```

@ . . . CLEAR

Propósito: Apagar (limpar) apenas uma área específica da tela.

Sintaxe: @ < Lin inicial >, < Col inicial > CLEAR

[TO<Lin final>,<Col final>]

Exemplo:

```
SET COLOR TO B+/W          / / muda a cor
CLS             // equivalente a CLEAR, ou seja limpa toda a
tela
SET COLOR TO W+/N            / / estabelece um novo padrão de
cor
@ 10,10 CLEAR TO 20,20               / / limpa uma região da
tela
@ 10,10 TO 20,20 DOUBLE           / / desenha uma moldura
(quadro)
```

@... SAY... GET

Propósito: Criar e executar um novo objeto GET (entrada de dados), colocando-o em exibição na tela.

Sintaxe: @ <linha>, < coluna> [SAY <exp> [<mascara SAY>]]
[WHEN<condição>]
[RANGE <inicial>,<final>]
[VALID <condição>]

Exemplo:

```
LOCAL VNOME :=SPACE(30) , VSALARIO : = 0    / / define inicia
variáveis
:
:
          // formata a digitação para maiúsculas
@ 12,10 SAY "NOME DO FUNCIONÁRIO.....:" GET VNOME PICT "@!"
          // edita os números no formato europeu
@ 14,10 SAY "SALÁRIO MENSAL:" GET VSALARIO PICT "@E 999,999,999.99"
VDATA := DATE( ) // cria a variável data contendo o DD/MM/AA
                 // contido no sistema operacional neste exemplo
                 // é assumida inicialmente a data do sistema para
                 // que o usuário não necessite preencher o campo,
                 // mas caso a data oferecida pelo programa não
                 // seja a correta basta que o usuário pressione
                 // qualquer tecla, que não sejam as teclas de
                 // movimentação, que a data é apagada, podendo
                 // assim o usuário escrever a data que desejar.
@ 16,10 SAY "ADMISSÃO..:" GET VADATA PICT "@K"
READ       // executa os gets pendentes
VENDERECO : = SPACE(35)
          // permite a edição do endereço, cujo tamanho é de
    // de 35 posições, em uma área da tela de apenas 20 posições,
          // rolando no sentido horizontal o que não couber no
20
          // espaços determinados por PICTURE "@s20".
@ 18,10 SAY "ENDEREÇO..:" GET VENDERECO PICTURE "@ 20"
READ       // executa o get pendente.
VCPF : = SPACE(14)
@ 10,15 SAY "C.P.F....:" GET VCPF PICTURE "999.999.999-9"
READ
VNOME : = SPACE(15)          // equivalente à picture "@!"
@ 11,15 SAY "NOME.....:" GET VNOME PICTURE "!!!!!!!!!!!!!!!"
```

```
VCODIGO := 0
     // os pontos serão editados, porem não serão gravados na
          // variável.
@ 12,15 SAY "CÓDIGO....:" GET VCODIGO PICTURE "@R 99.999.999"
READ
VALORI := 0
     // será aceito na digitação um valor que esteja compreendido
          // entre 0 e 1000.
@ 15,15 SAY "VALOR....:" GET VALORI PICT "9999" RANGE (0,1000)
READ
VALORII := 0
          // aceita apenas valores positivos
@ 16,50 SAY "VALOR....:" GET VALORII VALID (VALORII > 0)
READ
```

@... PROMPT

Propósito: Montar um menu de opções selecionáveis na tela.

Sintaxe: @ < linha >, < coluna >"< opção >" [MESSAGE <mensagem>]

Exemplo:

```
Local OPC := 1
SET WRAP ON     // habilita a rolagem da barra entre os extremos
                     // do menu
SET MESSAGE TO 23 CENTER   // determina a saída de mensagens da
                           // linha 23 da tela
DO WHILE .T.
CLEAR                // LIMPA A TELA
          //  cria  variáveis  para  facilitar  as  coordenadas  do
menu
L := 8
C := 32
          // montar a tela
@ 01,01 TO 24,79 DOUBLE
@ 02,02 TO 04,78
@ 03,01 SAY "ALT CONTROL INFORMÁTICA LTDA."
@ 03,60 SAY DATE( )
@ 03,70 SAY TIME( )
          // detalha o menu de barras
@ L,C      PROMPT "INCLUSÃO"     MESSAGE "INCLUSÃO DE DADOS"
@ L+1,C    PROMPT "ALTERAÇÃO"    MESSAGE "ALTERAÇÃO DE DADOS"
@ L+2,C    PROMPT "CONSULTA"     MESSAGE "CONSULTA DE DADOS"
```

```
@ L+3,C    PROMPT “EXCLUSÃO”      MESSAGE “EXCLUSÃO DE DADOS”
@ L+4,C    PROMPT “RELATÓRIOS”    MESSAGE “RELATÓRIOS DO SISTEMA”
@ L+5,C    PROMPT “UTILITÁRIOS”   MESSAGE “UTILITÁRIOS DO SISTEMA”
@ L+6,C    PROMPT “F I M”         MESSAGE “RETORNO AO DOS”
           // executa o menu e controla a barra
MENU OPC
DO CASE    // faca os casos
      CASE OPC = 1
           DO PROG1
      CASE OPC = 2
           DO PROG2
      CASE OPC = 3
           DO PROG3
      CASE OPC = 4
           DO PROG4
      CASE OPC = 5
           DO PROG5
      CASE OPC = 6
           DO PROG6
      CASE OPC = 7
           CANCEL          // cancela a execução do programa
ENDCASE
INKEY(0)              // aguarda QQ tecla
ENDDO
```

@. . . TO

Propósito: Desenha um quadro (moldura) a partir de coordenadas específicas da tela.

Sintaxe: @ <linhaI> , <colunaI > TO <linhaF>, <colunaF> [DOUBLE]

Exemplo:

```
SET COLOR TO B+/N
@ 10,10 CLEAR TO 20,20
@ 10,10 TO 20,20 DOUBLE
```

ACCEPT

Propósito: Cria uma entrada de dados via teclado e armazenar o conteúdo digitado em uma variável (tipo caracteres).

Sintaxe: ACCEPT [<mensagem de saída>] TO <var>.

Exemplo:

Local Vnome

```
CLEAR           // limpa a tela
ACCEPT "Digite o nome....:" TO VNOME
? "NOME QUE VOCÊ DIGITOU FOI....:", VNOME
```

APPEND BLANK

Propósito: Criar (inserir) um registro em branco no banco de dados aberto na área corrente de trabalho.

Sintaxe: APPEND BLANK

Exemplo:

Local Codvar, OP

```
/*
      NOME DO PROGRAMA: CADMULT1.PRG
            AUTOR : GORKI STARLIN
            FUNÇÃO: ESTE MODULO ANEXA DADOS NO ARQUIVO PAGAMENTO
*/
USE FOLHA INDEX CODX,NOMEX
DO WHILE .T.
            // lay out
CLEAR
      SET COLOR TO W+/N
      SET COLOR TO
      @ 01,01 TO 24,79 DOUBLE
      @ 02,02 TO 04,78
      @ 03,03 SAY "SÍRIOS INFORMÁTICA"
      @ 03,60 SAY ATE( )
      @ 03,70 SAY TIME( )
            // criar variáveis
CODVAR     = 0
SETORVAR   = 0
SALARIOVAR = 0
NOMEVAR    = SPACE(35)
CARGOVAR   = SPACE(15)
ATIVOVAR   = (.T.)
DATAVAR    = CTOD ("  /  /  ")
            // entrada de dados
```

```
@ 06,10 SAY "** CADASTRAMENTO DE FUNCIONÁRIOS **"
@ 08,10 SAY "CÓDIGO.....:" GET CODVAR PICTURE "9999"
READ
IF CODVAR = 0   // verifica se o usuário não digitou o código
      OP:= "S"    // cria a variável   OP
            @ 21,15 SAY "SAI DESTE MODULO.(S/N)..:" GET OP PICT "A"
            READ
            IF OP = "S"      // verifica a resposta do usuário
                  RETURN     // retorne
            ENDIF
      LOOP              // sobe a execução para linha do DO WHILE
ENDIF                   // fim do se
      SEEK CODVAR       // pesquisa no índice o conteúdo da variável
                        // CODVAR
IF EOF( )         // se não existe
      APPEND BLANK      // tenta criar um registro em branco
                        // entra com o restante dos dados do funcionário
@ 10,10 SAY "NOME FUNCIONÁRIO..:" GET NOMEVAR PICTURE "@!"
@ 12,10 SAY "SETOR TRABALHO....:" GET SETORVAR PICT "@9"
@ 14,10 SAY "CARGO FUNCIONAL...:" GET CARGOVAR PICT "@!"
@ 16,10 SAY "SALÁRIO.....:" GET SALARIOVAR PICT "9999999.99"
@ 18,10 SAY "FUNCIONÁRIO ATIVO.:" GET ATIVOVAR
@ 20,10 SAY "DATA ADMISSÃO.....:" GET DATAVAR
READ
            // grava os dados no registro em branco
      REPLACE COD WITH CODVAR
      REPLACE NOME WITH NOMEVAR
      REPLACE SETOR WITH SETORVAR
      REPLACE CARGO WITH CARGOVAR
       REPLACE ATIVO WITH ATIVOVAR
       REPLACE DTADM WITH DATAVAR
       REPLACE SALÁRIO WITH SALARIOVAR
      @ 21,20 SAY "** CADASTRO **"
      WAIT " "          // aguarda QQ tecla
      COMMIT            // atualiza fisicamente o registro
  ELSE              // se não
      @ 21,20 SAY "** REGISTRO JÁ CADASTRADO **"
      WAIT  " "         // aguarda QQ tecla
  ENDIF
ENDDO
```

APPEND FROM

Propósito: Anexa registro de um arquivo especificado para o arquivo que se encontra aberto na área corrente de trabalho.

Sintaxe: APPEND FROM [<escopo>] [FIELDS <campos>] [FROM <arquivo>]

```
[FOR <condição>] [WHILE <condição>]
[SDF/DELIMITED]
[WHITH BLANK / <delimitador>]
```

Exemplo:

```
USE FOLHA
APPEND FROM COPIAF FOR .NOT. DELETED( )    // copia apenas os
                                           //registros não marcados
?"termino da copia"
```

AVERAGE

Propósito: Calcular a média aritmética de campos ou expressões de arquivos de dados.

Sintaxe: AVERAGE <campos> TO <var's> [<escopo>] [FOR<condição>] [WHILE <condição>]

Exemplo:

```
USE FOLHA              // abre o arquivo de dados
AVERAGE SALÁRIO, COMISSÃO TO vcom     // calcula e armazena nas
                                      // variáveis
? "media salarial....:"+str(vsal)
? "media das comissões...:"+str(vcom = "A"   // calcula a media
                                // salarial, armazenando o
                                // resultado na variável VSAL,
                                // porém somente dos funcionários
                                // que trabalhem no setor A.
```

BEGIN SEQUENCE

Propósito: Define uma seqüência de comandos para uma BREAK.

Sintaxe: BEGIN SEQUENCE

... COMANDOS

[BREAK [<expressão>]]

... COMANDOS

[RECOUVER [USING <variável>]]

... COMANDOS

END [SEQUENCE]

Exemplo:

Local Contador, Intervalo

```
CONTADOR :=0
INTERVALO :=0
      DO WHILE CONTADOR < 50
            BEGIN SEQUENCE
                  CONTADOR++
                     IF CONTADOR > INTERVALO
                        BREAK CONTADOR
                     ENDIF
                  RECOUVER USING CONTADOR
                  ? "BLOCO DEFINIDO, CONTADOR ="+STR (CONTADOR)
                  INTERVALO+ = 5
            END SEQUENCE
                  ? "SAI FORA DO BEGIN SEQUENCE"
      ENDDO
            ? "LOOP TERMINADO"
```

CALL

Propósito: Executa uma rotina construída em outra linguagem de programação.

Sintaxe: CALL <rotina> WITH <parâmetros>

CANCEL

Propósito: Interromper a execução do programa que está sendo executado.

Sintaxe: CANCEL

CLEAR ALL

Propósito: Fecha todos os arquivos abertos e libera da memória todas as variáveis (Públicas e Privadas).

Sintaxe: CLEAR ALL

CLEAR GETS

Propósito: Libera todos os Gets pendente.

Sintaxe: CLEAR GETS

CLEAR MEMORY

Propósito: Libera todas as variáveis Públicas e Privadas da memória.

Sintaxe: CLEAR MEMORY

CLEAR SCREEN

Propósito: Limpa a tela sem liberar os Get's pendentes.

Sintaxe: CLEAR SCREEN

CLEAR TYPEAHEAD

Propósito: Libera todas as pendências de teclagens do Buffer (fila) do teclado.

Sintaxe: CLEAR TYPEAHEAD

Exemplo:

```
/* Neste exemplo antes de folhear o banco de dados com a função
BROWSE( ) é garantido que não existirá nenhuma pendência de
teclas do buffer do teclado, pois o mesmo será limpo através de
CLEAR TYPEAHEAD.*/
     BROWSE (5, 5, 23, 75)      // folheia os registros do B.D.
```

CLOSE

Propósito: Fechar arquivos, de qualquer tipo, que se encontrem devidamente abertos.

Sintaxe: CLOSE <área>< tipo>

Exemplo:

```
CLOSE ALL       // fecha todos os arquivos, de qualquer tipo
                // abertos em todas as áreas.
CLOSE folha INDEXES       // fecha todos os arquivos de índices
                          // que estiverem abertos na área
                          //(ALIÁS) FOLHA.
```

COMMIT

Propósito: Realiza a gravação em discos de todos os Buffers dos arquivos abertos.

Sintaxe: COMMIT

Exemplo:

```
/*
      NOME DO PROGRAMA: CADMONO1.PRG
            AUTOR : GORKI STARLIN
            FUNÇÃO: ESTE MODULO ANEXA DADOS NO ARQUIVO PAGAMENTO
*/
SET DATE TO BRIT        // põe as datas no formato DD/MM/AA
CLEAR
            // abre o arquivo e o incide
USE FOLHA INDEX CODX,NOMEX    // abre o arquivo de dados e o de índice
    DO WHILE .T.
CLEAR
  SET COLOR TO          //põe cor padrão
            // lay out
@ 01,01 TO 24,79 DOUBLE
@ 02,02 TO 04,78
@ 03,03 SAY "ALT CONTROL INF"
@ 03,60 SAY DATE( )
@ 03,70 SAY TIME( )
            // CRIAR VARIÁVEIS
CODVAR := SETORVAR := SALARIOVAR := 0
NOMEVAR     := SPACE(35)
CARGOVAR    := SPACE(15)
ATIVOVAR    := (.T.)
DATAVAR     := CTOD ("  /  /  ")
            // entrada de dados
@ 06,10 SAY "** CADASTRAMENTO DE FUNCIONÁRIOS **"
@ 08,10 SAY "CÓDIGO.....:" GET CODVAR PICTURE "9999"
READ
IF CODVAR = 0     // verifica se o usuário não digitou o código
      OP := "S"   // cria variável  OP
                  // pergunta se o usuário deseja sair do programa
```

```
            @ 21,15 SAY “SAI DESTE MODULO.(S/N)..:” GET OP PICT “A”
            READ
            IF OP = “S”        // verifica a resposta do usuário
                  RETURN       // retorne
            ENDIF
      LOOP              // sobe a execução para linha do DO WHILE
ENDIF                   // fim do se
      SEEK CODVAR       // pesquisa no índice o conteúdo da variável
                        // variável CODVAR
IF EOF( )         // se NÃO EXISTE
      // entra com o restante dos dados do FUNCIONÁRIO
@ 10,10 SAY “NOME FUNCIONÁRIO..:” GET NOMEVAR PICTURE “@!”
@ 12,10 SAY “SETOR TRABALHO....:” GET SETORVAR PICT “@9”
@ 14,10 SAY “CARGO FUNCIONAL...:” GET CARGOVAR PICT “@!”
@ 16,10 SAY “SALÁRIO...........:” GET SALARIOVAR PICT “9999999.99”
@ 18,10 SAY “FUNCIONÁRIO ATIVO.:” GET ATIVOVAR
@ 20,10 SAY “DATA ADMISSÃO.....:” GET DATAVAR
READ
            APPEND BLANK      // criar um registro em branco
            // grava os dados no registro em branco
      REPLACE COD WITH CODVAR
      REPLACE NOME WITH NOMEVAR
      REPLACE SETOR WITH SETORVAR
      REPLACE CARGO WITH CARGOVAR
      REPLACE ATIVO WITH ATIVOVAR
      REPLACE DTADM WITH DATAVAR
      REPLACE SALÁRIO WITH SALARIOVAR
            COMMIT      // salva todo o conteúdo do buffers de arquivos,
                        // armazenando-o em disco.
      @ 21,20 SAY “** CADASTRO **”
      INKEY(0)          // aguarda QQ tecla
  ELSE            // se não
      @ 21,20 SAY “** REGISTRO JÁ CADASTRADO **”
      INKEY(0)          // aguarda QQ tecla
  ENDIF     // fim do se
ENDDO       // fim do faça enquanto
```

CONTINUE

Propósito: Continua a pesquisa iniciada a partir do comando LOCATE.

Sintaxe: CONTINUE

Exemplo:

```
CLEAR
USE FOLHA   // abre o arquivo de dados
LOCATE FOR SETOR = “A”
      DO WHILE FOUND( )       // faça enquanto existir
```

```
        ? NOME, SALÁRIO, SETOR       // mostra os campos
        CONTINUE                     // continua a pesquisa
    ENDDO                            // fim do faça enquanto
```

COPY FILE

Propósito: Copiar o conteúdo de um arquivo, independente do seu tipo, para outro arquivo.

Sintaxe: COPY FILE <arquivo> TO <cópia>

Exemplo:

```
COPY FILE FOLHA.DBF TO FCOPIA.DBF
COPY FILE FOLHA.DBF TO FCOPIA.DBT
COPY FILE MENU.PRG TO A:MENU.PRG
```

COPY STRUCTURE

Propósito: Copiar apenas a estrutura do arquivo aberto na área corrente de trabalho.

Sintaxe: COPY STRUCTURE TO <copia> [FIELDS <campos>]

Exemplo:

```
USE FOLHA         // abre o arquivo de dados
COPY STRUCTURE TO TFOLHA           // cria o arquivo TFOLHA.DBF com a
                                   // mesma estrutura do arquivo FOLHA.DBF.
COPY STRUCTURE TO TFOLHA FIELDS NOME,SALARIO,COD     // cria o arquivo
                                   // TFOLHA contendo uma estrutura de apenas
                                                     // três campos.
```

COPY STRUCTURE EXTENDED

Propósito: Copia para outro arquivo informações referentes à estrutura de um arquivo de dados aberto.

Sintaxe: COPY STRUCTURE EXTENDED TO <arquivo>

Exemplo:

```
USE FOLHA         // abre o arquivo de dados FOLHA.DBF
COPY STRUCTURE EXETENDED TO EFOLHA       // copia sua estrutura para o
                                         // arquivo EFOLHA.DBF
USE EFOLHA        // abre o arquivo contendo a estrutura de FOLHA.DBF
```

```
LIST FIELD_NAME, FIELD_TYPE, FIELD_LEN, FIELD,DECX          // lista os
                                                            // registros
```

COPY TO

Propósito: Copia registros de bancos de dados (.DBF) para outro arquivo (.DBF ou no formato ASCII).

Sintaxe: `COPY TO [ FIELDS <campos> ] TO <arquivo> [<escopo>] [FOR <condição>] [SDF/DELIMITED [ WITH BLANK / delimitador]`

Exemplo:

```
USE FOLHA          // abre o arquivo FOLHA.DBF
COPY TO CFOLHA          // copia os registros para o arquivo CFOLHA.DBF
COPY TO FOLHA FOR SETOR = "A"       //   somente   serão   copiados   os
registros
                                    // que possuem a letra A inicial no
                                    // campo SETOR
COPY TO CFOLHA RECORD 3             // é copia apenas o registro 3
COPY TP CFOLHA DELIMITED            // copia para o arquivo CFOLHA.TXT
                                    // no formato delimitado
TYPE CFOLHA.TXT               // mostra o conteúdo do arquivo CFOLHA.TXT
```

COUNT

Propósito: Calcular o totalizante referente à quantidade de registos.

Sintaxe: COUNT TO <var> [<escopo>] [FOR <condição>] [WHILE <condição>]

Exemplo:

```
USE FOLHA
COUNT TO RESULTADO
? RESULTADO
COUNT TO RESULTADO2 FOR SETOR = "A"
COUNT TO RESULTADO3 FOR SETOR = "A" .AND. CARGO = "ESCRITURARIO"
?RESULTADO, RESULTADO2, RESULTADO3
```

CREATE

Propósito: Criar um arquivo de estrutura (.DBF) vazio.

Sintaxe: CREATE <arquivo>

Exemplo:

```
CREATE TESTRU          // cria o arquivo de estrutura
APPEND BLANK           // cria um registro em branco para descrição de
                       // um campo da estrutura.
REPLACE ;                          // define o:
      FIELD_NOME WITH "COD"    ;     // nome do campo
      FIELD_TYPE WITH "C"      ;     // tipo do campo
      FIELD_LEN WITH 5         ;     // tamanho do campo
      FIELD_LEN WITH 0         ;     // número de casas decimais
CLOSE // fecha o arquivo de estruturas
CREATE FOLHA TESTRU          //  declara  o  comando  CREATE  FROM  para
criar
                             //  um  novo  arquivo  .DBF  a  partir  do
arquivo
                             // de estrutura TESTRU
DIR *.DBF         // mostra todos os arquivos .DBF do diretório
```

CREATE FROM

Propósito: Criar um arquivo de dados (.DBF) a partir de um arquivo de estruturas.

Sintaxe: CREATE FROM <novo> FROM <arquivo_estrutura>

Exemplo:

```
CREATE TESTRU          // cria o arquivo de estrutura
APPEND BLANK           // cria um registro em branco para descrição de
                       // um campo da estrutura.
REPLACE ;                          // define o:
      FIELD_NOME WITH "COD" ;  // nome do campo
      FIELD_TYPE WITH "C"   ;  // tipo do campo
      FIELD_LEN WITH 5      ; // tamanho do campo
      FIELD_LEN WITH 0      ; // numero de casas decimais
CLOSE // fecha o arquivo de estruturas
CREATE FOLHA FROM TESTRU           //  declara  o  comando  CREATE  FROM
para
                             // criar um novo arquivo .DBF a partir do
                             // arquivo de estrutura TESTRU
DIR *.DBF         // mostra todos os arquivos .DBF do diretório
```

DECLARE

Propósito: Declara variáveis ou vetores privates no programa.

Sintaxe: DECLARE <identificador> [:= <valor>]

DELETE

Propósito: Marcar um registro para ser apagado.
Sintaxe: DELETE <escopo> [FOR <condição>]
[WHILE <condição>]

Exemplo:

```
USE FOLHA
DELETE ALL              // marca TODOS os registros
DISPLAY ALL NOME, SALÁRIO, COD          // mostra os registros
INKEY(0)
SET DELETE ON           // filtra os registros marcados
DISPLAY ALL NOME, SALÁRIO, COD          // mostra os registros
INKEY(0)
RECALL ALL              // recupera todos os registros
DISPLAY ALL NOME, SALÁRIO, COD          // mostra os registros
INKEY(0)
DELETE FOR SETOR = "A"        // marca os funcionários do setor A
DISPLAY ALL NOME, SALÁRIO, COD          // mostra os registros
? "FIM"
```

DELETE FILE

Propósito: Apagar um arquivo, de qualquer tipo, do disco.
Sintaxe: DELETE FILE <arquivo>

Exemplo:

```
IF FILE ("FOLHA.DBF")         se existir FOLHA.DBF
      DELETE FILE FOLHA.DBF
      ? "ARQUIVO FOI APAGADO"
ENDIF
DIR *.DBF         // mostra todos os arquivos com a extensão .DBF
```

DIR

Propósito: Mostra a lista dos arquivos contidos em um diretório.
Sintaxe: DIR [<drive>] [<caminho>] [<máscara>]

Exemplo:

```
DIR         // mostra todos os arquivos (BDF) e seus dados
DIR *.*     // mostra todos os arquivos do diretório
DIR *.prg   // mostra todos os programas do diretório
DIR a: *.*  // mostra todos os arquivos do diskete do drive A
```

DISPLAY

Propósito: Mostra registros de um arquivo de dados na console.

Sintaxe:
```
DISPLAY <campos> [TO PRINTER]
[TO FILE <nome_arquivo>]
[OFF]  [<escopo>]  [FOR <condição>]
[WHILE <condição>]
```

Exemplo:

```
USE FOLHA         // abre o arquivo de dados
DISPLAY COD, NOME, SALÁRIO ALL           // mostra todos os registros
DISPLAY COD, NOME, SALÁRIO         //   mostra   somente   o   registro
corrente
DISPLAY COD, NOME, SALÁRIO ALL FOR SETOR = “A”     //      mostra      os
registros
                                              // dos funcionários que
                                          // que trabalham no setor A
```

DO

Propósito: Executa um programa ou um procedimento.

Sintaxe: DO <nome> [WITH <lista de parâmetros>]

Exemplo:

```
:
      IF OP = 2
            DO PROG1
      ELSEIF OP =3
            DO PROG2
      ELSE
            DO PROG4 WITH NOME
      ENDIF
:
:
```

DO CASE

Propósito: Criar uma estrutura de testes condicionais, onde apenas uma é executada.

Sintaxe:

```
DO CASE
      CASE <condição>
            . . . . instruções
      [CASE <condição2>]
            . . . . instruções
      [OTHERWISE]
            . . . . instruções
END[CASE]
```

Exemplo:

```
DO CASE
      CASE OP = 2
            DO PROG1
      CASE OP = 3
            DO PROG2
      OTHERWISE
            RETURN
ENDCASE
```

DO WHILE

Propósito: Executa uma estrutura de controle enquanto uma condição for verdadeira.

Sintaxe:

```
DO WHILE <condição>
            . . . . <instruções>
      [EXIT]
            . . . . <instruções>
      [LOOP]
            . . . . <instruções>
END[DO]
```

Exemplo:

```
      :
      :
VARSAI :=  " "
   ┌─→ DO WHILE VARSAI .NOT. $ "SN"        // faça enquanto VARSAI não
   │                                       // contiver "S" ou "N"
   │
   │
   └
```

```
            // pergunta dirigida ao operador
        @ 21,20 SAY "SAIR DESTE MODULO (S/N)..:" GET VARSAI PICT "!"
        READ
ENDDO                   // fim do faça enquanto
:
:
```

EJECT

Propósito: Avança a página da impressora posicionando a cabeça de impressão no local de inicialização da próxima página.

Sintaxe: EJECT

Exemplo:

```
LOCAL L, PG
USE FOLHA
L:= 0       // inicializa uma variável para controle da quantidade de
            // linhas impressas
PG:= 0
GO TOP      // vá para o inicio do arquivo
SET PRINT ON            // liga a saída comum para a impressora
SET CONSOLE OFF         // desabilita a saída da console
   ┌──→ DO WHILE .not. EOF( )         // faça enquanto não fim do arquivo.
   │       IF L = 0 .OR. L=60         // se L for 0 ou 60
   │              EJECT
   │              PG++    // acumula +1 na variável
   │              ? "RELATÓRIO DE FUNCIONÁRIOS"
   │              ?
   │              ? "Pagina:"+str(pg)
   │              replicate ("=", 78)       // traça uma linha
   │              1 := 7
   │           ENDIF
   │           ? COD, NOME, SALÁRIO         // imprime os campos
   │           SKIP        // pule para o próximo registro
   │           L++
   └──── ENDDO              // fim do faça enquanto
         :
         :
```

ERASE

Propósito: Apagar um arquivo, de qualquer tipo, do disco.

Sintaxe: ERASE <arquivo>

Exemplo:

```
IF FILE ("FOLHA.DBF")          // se existir FOLHA.DBF
      ERASE FOLHA.DBF
      ? "ARQUIVO FOI APAGADO"
ENDIF
DIR *.DBF         // mostra todos os arquivos com a extensão . DBF
```

EXTERNAL

Propósito: Declarar uma lista de símbolos ou rotinas externas para o linker.

Sintaxe: EXTERNAL <lista>

Exemplo:

```
EXTERNAL funções
      :
      :
```

EXIT PROCEDURE

Propósito: Declara um procedimento de saída.

Sintaxe: EXIT PROCEDURE <nome da rotina/procedimento>
[FIELDS <lista de símbolos> [IN <alias>]]
[MENVAR <lista de símbolos>]
:
<expressões executáveis>
:
[return]

Exemplo:

```
// COMPILE ESTE PROGRAMA COM /N
ANNOUNCE MEUSYSTEMA
STATIC nSEGUNDOS
PROCEDURE PRINCIPAL( )
nSEGUNDOS := SECONDS( )
AEVAL (ASORT (DIRECTORY ( "*.*")),;
           {|Anomes | QQUT (Anomes[1} ) } )
return // termina o programa.
```

```
EXIT PROCEDURE SAÍDA( )              / / rotina de saída do programa
?
? "TEMPO: "
?? SECONDS ( ) - nSEGUNDOS
RETURN                              / / finaliza definitivamente
```

FIELD

Propósito: Especifica nomes de campos de arquivos de dados (.DBF).

Sintaxe: FIELD <lista [IN <apelido>]

Exemplo:

```
FIELD NOME,COD,SALARIO INTO FOLHA
FIELD CODCARGO,CREDITOS INTO CARGOS
USE FOLHA ALIAS FOLHA
USE CARGOS ALIAS FOLHA
      <instruções>
      :
      :
? cod,codcardi          // equivalente a FOLHA-> COD,CARGOS -> CODCARGO
? nome                  // equivalente a folha -> nome
      :
      :
```

FIND

Propósito: Pesquisa no primeiro índice, o registro que possua uma chave especificada.

Sintaxe: FIND <string>

Exemplo:

```
USE FOLHA INDEX CODX, NOME // abre o arquivo de dados folha.dbf
                           // e seus respectivos arquivos de
                           // índices CODX, NOMEX
FIND "3020"                // pesquisa o código = 3020
     IF FOUND( )                 // se existir
          DISPLAY COD,NOME,SALARIO
     ENDIF
CODVAR := SPACE(4)
@ 10,20 SAY "DIGITE O CÓDIGO...:" GET CODVAR PICTURE "9999"
READ
FIND CODVAR             // pesquisa o conteúdo da variável
```

```
IF FOUND( )          // se existir
     DISPLAY COD,NOME,SALARIO
ENDIF
```

FOR. . . NEXT

Propósito: Executa uma estrutura de controle, um determinado número de vezes.

Sintaxe:

```
FOR <contador> = <inicio> TO <fim> STEP
<passo>
   ..... <instruções>
  [EXIT]
   ..... <instruções>
  [LOOP]
NEXT
```

Exemplo:

```
LOCAL TREGISTROS
USE CADASTRO
COUNT TO TRESGISTROS
GO TOP
FOR I = 1 TO TREGISTROS STEP 1
   DISPLAY NOME, ENDEREÇO, TEL   // exibe o registro corrente
   SKIP         // pula para o próximo registro
NEXT
? ”FIM”
```

FUNCTION

Propósito: Cria (declara) uma função definida pelo usuário (UDF).

Sintaxe:

```
[STATIC] FUNCTION <FUNÇÃO> [(PARAMENTRO1,..)]
[LOCAL <identificador>,...]
[FIELD <lista de identificador> [IN <apelido>]
MEMVAR <lista de identificadores>
   :
   :
<instruções>
   :
   :
RETURN [<informação>]
```

Exemplo:

```
LOCAL VAR1, VAR2, VAR3, X
var1 := 3
var2 := 7
var3 := 100
      :
? soma (var3,var2)    // resultado : 107 (na tela)
? soma (var1,var2)    // resultado : 10 (na tela)
x:= soma(var3,300)    // resultado : 400 (na variável)
      :
      :
      :
FUNCTION SOMA ( P1, P2 )   // declara a função e recebe os
                           // parâmetros
R := P1+P2       // soma os parâmetros
RETURN R         // retorna a execução para rotina que chamou
                 // acompanhada do valor contido na variável R,
```

GO

Propósito: Desloca o ponteiro interno do arquivo de dados para um determinado registro.

Sintaxe: GO [TO] <registro> | BOTTOM | TOP

Exemplo:

```
USE FOLHA
GO 6       // vá para o registro (record) numero 6
DISPLAY NOME, COD, SALÁRIO
GO TOP     // vá para o inicio do arquivo
DISPLAY NOME, COD, SALÁRIO
GO BOTTOM  // vá para o fim do arquivo
DISPLAY NOME,COD,SALARIO
```

IF

Propósito: Executa instruções somente quando uma expressão condicional for verdadeira.

Sintaxe:

```
IF <condição>
   <instruções>
[ELSEIF < condição2>
   <instruções>
[ELSE]
```

```
        <instruções>
    END[IF]
```

Exemplo:

```
LOCAL MEDIA:= 0
CLEAR
@ 10,10 SAY "DIGITE A MEDIA DO ALUNO...:"GET MEDIA
READ
     IF MEDIA <5
          ?   "REPROVADO"
     ELSEIF MEDIA = 5
          ?   "RECUPERAÇÃO"
     ELSE
          ?   "APROVADO"
     ENDIF
     :
     :
```

INIT PROCEDURE

Propósito: Especificar uma procedure que será executada antes da primeira rotina do Programa.

Sintaxe:

```
INIT PROCEDURE <nome da rotina/procedimento>
[FIELDS <lista de símbolos> [IN <alias>]]
     [LOCAL <símbolos> [:= valor]]
     [MEMVAR <lista de símbolos>
          <expressões executáveis>
          :
          [Return]
```

Exemplo:

```
// COMPILE ESTE PROGRAMA COM /N
ANNOUNCE MEUSYSTEMA
STATIC nSEGUNDOS
PROCEDURE PRINCIPAL( )
     AEVAL   (ASCOL (DIRECTORY ("*.*") ) ,;
               { | Anomes | QOUT ( Anomes [1] ) } )
     RETURN       // termina o programa.
INIT PROCEDURE INICIAL( )       // rotina de inicialização
nSEGUNDOS := SECONDS( )
RETURN
EXIT PROCEDURE SAÍDA( )         // rotina de saída do programa.
```

```
  ?
  ?  "TEMPO:"
  ?? SECONDS() - nSEGUNDOS
  RETURN        // finaliza definitivamente
```

INDEX

Propósito: Criar um arquivo de índice (.NTX) para um determinado banco de dados (.DBF)

Sintaxe: INDEX ON <chave> TO <índice> [UNIQUE] [FOR <Condição>]

Exemplo:

```
USE CADASTRO
CLEAR
? "INDEXANDO"
INDEX ON NOME TO INDICE1        // indexa o arquivo pelo nome e
                      // cria o arquivo que conterá o controle de
                                // índice INDICE1.NTX
LOCAL VNOME:= SPACE(30)
@ 10,10 SAY "DIGITE O NOME..:" GET VNOME PICTURE "@!"
READ
? "PESQUISANDO"
SEEK VNOME
IF FOUND( )           // se existir
     DISPLAY NOME, ENDEREÇO, CIDADE       // mostra o registro
ENDIF
? "REGISTRO NÃO ENCONTRADO"
```

INPUT

Propósito: Realizar a entrada de dados de um expressão e armazena a mesma em uma variável.

Sintaxe: INPUT [<mensagem>] TO <variável>

Exemplo:

```
LOCAL VAR
CLEAR
INPUT "DIGITE QUALQUER COISA..:" TO VAR
```

```
? "VOCÊ DIGITOU...:"
?? VAR
```

JOIN

Propósito: Criar um novo arquivo a partir de outros dois.

Sintaxe: `JOIN WITH <2° arquivo> TO <novo arquivo>`
`FOR <condição> [FIELDS <lista de campos>]`

Exemplo:

```
USE VENDAS            // possui os campos cod_vend, cod_produto e
                      // valor
USE CADVENDEDOR new         // possui os campos cod_vend, nome
JOIN WITH VENDAS TO COMISSÃO FOR COD_VEND= VENDAS -> COD_VEND;
               FILEDS COD_VEND, NOME, VALOR
          // será criado o arquivo COMISSÃO.DBF com os registros
          // lidos dos arquivos e a estrutura deste arquivo será
          // os campos declarados após o argumento FILEDS.
```

KEYBOARD

Propósito: Preencher o buffer do teclado com uma expressão caractere.

Sintaxe: KEYBOARD <expressão caractere>

Exemplo:

```
KEYBOARD "a"
KEYBOARD  CHR(65)          // resultado:   A
KEYBOARD  CHR(130)         // resultado:   é
```

LABEL FORM

Propósito: Executa a saída de etiquetas a partir de um arquivo do formato. .LBL.

Sintaxe: `LABEL FORM <arquivo.LBL> [TO PRINTER]`
`[TO FILE]`

```
                    [<ESCOPO>]    [SAMPLE]         [WHILE
<condição>]
                    [FOR<condição>]
```

Exemplo:

```
USE  MALA  INDEX  NOME
LABEL FORM ETIQUETAS TO PRINTER SAMPLE    // imprime as etiquetas
```

LIST

Propósito: Lista os registros de arquivos de dados.

Sintaxe:
```
LIST<lista exp>        [TO PRINTER]
[TO FILE   <arquivo>]
[<escopo>]        [WHILE<condição>]
[FOR <condição>]
[OFF]
```

Exemplo:

```
USE  MALA
LIST  NOME,  ENDEREÇO,  CIDADE
LIST  NOME,  ENDEREÇO,  CIDADE  TO   PRINTER    // lista impressa
```

LOCATE

Propósito: Localizar um registro dentro do banco de dados.

Sintaxe: LOCATE [<escopo>] FOR <condição> WHILE <condição>

Exemplo:

```
USE   FOLHA
LOCATE  FOR NOME =“João”
IF FOUND()      / / se existir
     DISPLAY  NOME,  SALÁRIO,  SETOR
ELSE
     ? “não  localizado”
ENDIF
```

LOOP

Propósito: Saltar a execução do programa para a linha DO WHILE, ou FOR.

Sintaxe: LOOP

LOCAL

Propósito: Declarar uma variável ou matriz como local.

Sintaxe: LOCAL<identificador> [:= <inicializador>],...

Exemplo:

```
LOCAL VAR,VAR2:= 10    //    declara as variáveis como
locais
? VAR2
LOCAL MATRIZ1 [30] [10]    // declara a matriz como local
```

MEMVAR

Propósito: Declara nomes de variáveis de memória Privadas ou Públicas.

Sintaxe: MEMVAR <lista de variáveis>

Exemplo:

```
USE MALA
MEMVAR NOME          // declara como sendo variáveis de memória
LOCAL NOME           // declara como sendo uma variável de
                     // memória local
     :
? NOME               //  mostra o conteúdo da variável nome
? MALA →NOME         // mostra o conteúdo do campo nome
```

MENU TO

Propósito: Executa um menu de barras luminosas.

Sintaxe: MENU TO <variável>

NOTE

Propósito: Cria uma linha de comentário dentro do programa.
Sintaxe: NOTE <texto>

Exemplo:

```
NOTE esta linha não será copilada, ou seja e apenas um
NOTE       comentário
? "esta linha é uma instrução que será e apenas  será  copilada"
// esta linha também é um comentário
&& também é um comentários
/* estas linhas também são comentários     */
```

PACK

Propósito: Remove (apaga) fisicamente registros marcados para deleção.
Sintaxe: PACK

Exemplo:

```
USE  MALA  INDE  NOME
PACK       // remove fisicamente do arquivo os registros marcados
```

PARAMETER

Propósito: Criar variáveis de memória para o recebimento de parâmetros.
Sintaxe: PARAMETER <lista de variáveis>

Exemplo:

```
MENSAGEM  (5,  5, "OI !")
FUNCTION  MENSAGEM( )
PARAMETER  LINHA,  COLUNA,  DADO    //recebe  valores  da  rotina
                                  // que chamar  esta  função
@  LINHA,  COLUNA  SAY  DADO
RETURN  NIL
```

PRIVATE

Propósito: Cria e inicializa variáveis ou matrizes como sendo privadas.

Sintaxe: PRIVATE <identificador>[:= <inicializador>],

Exemplo:

```
PRIVATE  MATRIZ1  [20] [30]     //  declara   que  a  matriz
                                //  será  privada
PRIVATE  A,  B,  C              // declara que as variáveis são
                                // privadas
A:=8                            //  atribui  um  valor  a
                                // variável
PRIVATE  DATA:=DATE(  )         //  declara  e  inicializa  a
                                // variável privada
```

PROCEDURE

Propósito: Cria um procedure e seus parâmetros.

Sintaxe:

```
[STATIC] PROCEDURE <procedure> [(lista parâmetros)]
[FIELD <lista de campos>[IN <apelidos>]]
[LOCAL
<identificador>
[:= <inicializador>],,,]
[MEMVAR <lista de identificadores>]
[STATIC <identificador>
                                                [:=
<inicializador],,,]
:
<instruções>
:
[RETURN]
```

Exemplo:

```
      :
      :
      :
MENSAGEM(20,10,"NÃO  ENCONTRADO")
      :
      :
PROCEDURE  MENSAGEM(LINHA,  COLUNA,  DADO)
@  LINHA,  COLUNA  SAY  DADO
RETURN
```

PUBLIC

Propósito: Cria e inicializa variáveis e matrizes públicas.

Sintaxe: PUBLIC <identificador>[:= <inicializador>],,,

Exemplo:

```
PUBLIC  MATRIZ3 [48] [10]        //     define   a   matriz   como
publica
PUBLIC  A,  B,  C             //define as variáveis como públicas
      :
      :
A:=10 // inicializa  a  variável
```

QUIT

Propósito: Termina a execução do programa.

Sintaxe: QUIT

Exemplo:

```
      :
      RESPOSTA:="S"
      @ 20,10 SAY "SAIR DESTE PROGRAMA...:" GET RESPOSTA  PICT "!"
      READ
            IF  RESPOSTA = "S"
                  QUIT       // termina  o programa
            ELSE
                  LOOP //sobe a execução para linha de DO  WHILE
            ENDIF
      :
```

:

READ

Propósito: Executar edição das variáveis especificadas pelo comando @.. SAY.. GET.

Sintaxe: READ[SAVE]

Exemplo:

LOCAL VNOME, VENDEREÇO, VSALÁRIO

```
      VNOME:= SPACE(30)
      VENDEREÇO:= SPACE(35)
      VSALÁRIO:= 0.00
@ 10,10 SAY "DIGITE O NOME...:" GET  VNOME  PICT "!"
@ 12,10 SAY "DIGITE O ENDEREÇO..:" GET VENDEREÇO
@ 14,10 SAY "DIGITE O SALÁRIO...:" GET VSALÁRIO PICT "@E  9,999.99"
READ          //  executa  e  no  final  libera  os  três  GET's
pendentes
```

RECALL

Propósito: Recupera registros marcados para a eliminação através do comando DELETE.

Sintaxe: RECALL <escopo> [WHILE<condição>]

[FOR<condição>]

Exemplo:

```
USE  MALA
GOTO  3
IF  DELETED(  ) // se o registro se encontra marcado
                // (deletado)
RECALL          //  recupere
ENDIF
```

REINDEX

Propósito: Recriar os arquivos de índices abertos nas áreas de trabalho corrente.

Sintaxe: REINDEX
[EVAL<Condição>]
[EVERY<nRegistro>]

Exemplo:

```
USE MALA INDEX INOME, ICOD
REINDEX / / reorganiza os arquivos
INOME, ICOD
:
:
```

RELEASE

Propósito: Libera da memória várias Públicas e Privadas.

Sintaxe: RELEASE <lista de variáveis>
[ALL [LIKE / EXCEPT <eskeleto>]]

Exemplo:

```
RELEASE ALL LIKE V*     // libera todas as variáveis que
                        // começam com a letra V
RELEASE VNOME           // libera a variável VNOME
```

RENAME

Propósito: Renomear um arquivo

Sintaxe: RENAME <nome atual> TO <novo nome>

Exemplo:

```
RENAME ARQ.TXT TO ARQ_NOVO.TXT     // troca o nome do arquivo
RENAME MALA.DBF TO POSTAL.DBF
```

REPLACE

Propósito: Substituir o conteúdo de um campo por uma expressão.

Sintaxe: REPLACE <campo> WITH <expressão>
[FOR <Condição>]
[WHILE <condição>]

Exemplo:

```
USE  MALA  INDEX  ICOD
APPEND  BLANK              // cria  um  registro  em  branco
REPLACE  COD  WITH  23,  NOME  WITH  "JOÃO"
                     // preenche  os  campos
     :
     :
```

REPORT FORM

Propósito: Realizar a saída de um relatório para console ou impressora.

Sintaxe: REPORT FORM <nome do arquivo> [<escopo>]
[TO PRINTER]
[TO FILE <nome>] [FOR <Condição>]
[WHILE <Condição>]
[PLAIN] [HEADING <cabeçalho>] [NOEJECT]
[SUMMARY]

Exemplo:

```
USE FOLHA INDEX INOME
REPORT FORM REL1 TO PRINTER     // relatório impresso dos
                                   // registros
REPORT FORM REL1 TO PRINTER HEADING "ALT CONTROL - SETOR  4" ;
FOR SETOR = 4     // imprime somente os funcionários do  setor  4
```

REQUEST

Propósito: Declara módulos a serem chamados.

Sintaxe: <módulos>.

RESTORE

Propósito: Carregar variáveis gravadas de um arquivo (.mem) do disco.

Sintaxe: RESTORE <nome do arquivo> [ADDITIVE]

Exemplo:

```
A:=4
NOME:="JOÃO"
SAVE TO ARQVAR// salva todas as variáveis de memória
                // no arquivo ARQVAR.MEM
RELEASE ALL    // apaga todas as variáveis
RESTORE FROM ARQVAR     // restaura as variáveis do arquivo
ARQVAR.MEM
? A
? NOME
```

RESTORE SCREEN

Propósito: Restaurar no vídeo uma tela salva anteriormente.

Sintaxe: RESTORE SCREEN [FROM <tela>]

Exemplo:

```
CLEAR
@ 10,10 TO 23,79
@ 15,15 SAY "ESTA TELA SERÁ SALVA"
SAVE SCREEN TO IMAGEM
INKEY(0)       / / aguarda uma tecla
CLEAR     / / limpa a tela
RESTORE SCREEN FROM IMAGEM        // recupera a tela
                              // gravada na variável imagem
```

RETURN

Propósito: Terminar a execução de uma procedure, programa ou função do usuário.

Sintaxe: RETURN <valor>

Exemplo:

```
? SITUAÇÃO (3,7,8,10)
FUNCTION SITUAÇÃO(N1, N2, N3, N4)
MÉDIA :=(N1+N2+N3+N4)/4
IF MÉDIA = >6
RETURN "APROVADO"
ELSE
RETURN "REPROVADO"
ENDIF
```

RUN

Propósito: Executar um programa ou comando do sistema operacional.

Sintaxe: RUN <descrição>

Exemplo:

```
? "FAVOR ATUALIZAR A HORA DO SISTEMA!."
? "FAVOR ATUALIZAR A DATA DO SISTEMA!."
! DATE
```

SAVE

Propósito: Salvar em um arquivo no disco, variáveis de memória e seus conteúdos.

Sintaxe: SAVE TO <arquivo> [ALL[LIKE|EXCEPT <esqueleto>]]

Exemplo:

```
A:=9
VNOME := "JOÃO"
VENDE:= "RUA DAS CAMÉLIAS 44"
```

```
SAVE TO ARQVAR2 ALL LIKE V*     //  salva:  VNOME  E  VENDE  no
arquivo
                                // ARQVAR2.MEM
SAVE TO ARQVAR             // salva todas as variáveis no arquivo
                           // ARQVAR.MEM
```

SAVE SCREEN

Propósito: Salvar a tela atual no buffer ou em uma variável

Sintaxe: SAVE SCREEN [TO <tela>]

SEEK

Propósito: Pesquisar nos registros do banco de dados indexado uma chave especificada.

Sintaxe: SEEK <chave>

Exemplo:

```
USE MALA INDEX INOME
SEEK "JOÃO"           // Equivalente A: DBSEEK ("JOÃO")
     IF FOUND( )           // se existir
          DISPLAY NOME, ENDEREÇO, CIDADE
     ELSE
          ? "NÃO ENCONTRADO"
     ENDIF
```

SELECT

Propósito: Seleciona uma área de trabalho.

Sintaxe: SELECT <Nome da área>|<apelido>

Exemplo:

```
USE MALA INDEX INOME
SELECT 0        // seleciona o próxima área disponível
USE FOLHA INDEX CODF
LIST NOME, SALÁRIO, SETOR, COD
```

```
SELECT MALA     // seleciona o arquivo área MALA
LIST COD, CLIENTE, CIDADE
LIST MALA → CLIENTE, FOLHA → SALÁRIO// lista registro de
                                        // outra área
```

SET ALTERNATE

Propósito: Realiza a saída do console para um arquivo (ASCII) a ser gravado no disco.

Sintaxe: SET ALTERNATE TO <arquivo> |[ON]|[OFF]|<(.T.)/(.F.)>

Exemplo:

```
SET ALTERNATE TO ARQSAIDA.TXT
AET ALTERNATE ON     // lida a saída para o arquivo
USE MALA INDEX ICEP
LIST CLIENTE, CIDADE, ESTADO
SET ALTERNATE OFF          // suspende a saída para o arquivo
CLOSE ALTERNATE            // fecha a operação com o arquivo
                          // alternativo.
TYPE ARQSAID.TXT
```

SET BELL

Propósito: Controla a saída sonora na operação de entrada de dados.

Sintaxe: SET BELL ON|OFF|<(.T.)/(.F.)>

SET CENTURY

Propósito: Possibilita configurar os dígitos dos séculos das datas.

Sintaxe: SET CENTURY ON|OFF|<(.T.)/(.F.)>

Exemplo:

```
SET DATE TO BRIT      // escolher o formato da data
? date( )                  // resultado: DD/MM/AA
SET CENTURY ON             // configura as datas para quatro
                           // dígitos no ANO
? date( )                  // resultado: DD/MM/AAAA
SET CENTURY OFF            // retorna ao padrão
```

SET COLOR

Propósito: Definir as cores que serão exibidas na tela.

Sintaxe: SET COLOR TO [<padrão>, <destaque>, <borda>, <fundo>, <não selecionado>] | <string>

Exemplo:

```
VNOME := SPACE(30)
PADRAO1 := "W/N, N/N"
PADRAO2 := "B/N, N/W"
SET COLOR TO (PADRAO1)
@ 10,10 SAY "DIGITE O NOME...:" GET VNOME PICTURE "@!"
SET COLOR TO (PADRAO2)
READ
SET COLOR TO W+,B
? "VOCÊ DIGITOU O NOME...:"
?? VNOME
```

SET CONFIRM

Propósito: Configurar a confirmação de entrada de dados de GET's.

Sintaxe: SET CONFIRM ON|OFF|<(.T.)/(.F.)>

Exemplo:

```
CLEAR
LOCAL VNOME := SPACE(15)
@ 10,10 SAY "DIGITE O SE NOME POR COMPLETO...:" GET VNOME
READ
SET CONFIRM ON        // liga a confirmação
@ 20,10 SAY "DIGITE O SEU NOME POR COMPLETO...:" GET VNOME
READ
```

SET CONSOLE

Propósito: Configurar a saída do console
Sintaxe: SET CONSOLE ON|OFF

SET CURSOR

Propósito: Configurar o formato da edição de campos ou variáveis do tipo Data.
Sintaxe: SET DATE [TO] <nome>

Exemplo:

```
SET DATE TO ITALIAN
? "A DATA DE HOJE E....:"
?? DATE( )
SET DATE TO GERMAN
VDATA:=CTOD ("  /  /  ")
@ 10,10 SAY "DIGITE QUALQUER DATA...:" GET VDATA
READ
SET DATE TO ANSY
? "Mudando o formato da data"
? "A data que você digitou foi...:"
?? VDATA
```

SET DECIMALS

Propósito: Configurar a quantidade de casas decimais exibidas.
Sintaxe: SET DECIMALS <quantidade de decimais>

Exemplo:

```
SET FIXED ON
SET DECIMALS TO 2          // 2 casas decimais (o padrão)
? 10/3
? 20/7
SET DECIMALS TO 5
? 10/3
? 20/7
```

SET DEFAULT

Propósito: Configurar a unidade de disco em que os arquivos serão processados.

Sintaxe: SET DEFAULT TO <disco\diretório\ , , ,>

Exemplo:

```
SET DEFAULT TO A:     // muda a leitura de arquivo para o diskete
SET DEFAULT TO C:\CLIPPER5      // muda para a unidade C no
                                // diretório \ CLIPPER5
```

SET DELETED

Propósto: Ativar ou desativar os registros marcados para eliminação.

Sintaxe: SET DELETED ON|OFF|(.T.)/(.F.)

SET DELIMITERS

Propósito: Ativar ou desativar a edição de caracteres que serão utilizados como delimitadores de GET's.

Sintaxe: SET DELIMITERS ON|OFF|(.T.)/(.F.)

SET DELIMITER TO

Propósito: Define delimitadores para edições GET's.

Sintaxe: SET DELIMITERS TO <delimitadores> [DEFAULT]

Exemplo:

```
CLEAR
VNOME:= VENDERECO:= SPACE(30)
SET DELIMITER ON      // liga a edição de delimitadores
SET DELIMITER TO "::"       // estabelece novos delimitadores
@ 10,10 SAY "DIGITE O NOME...:" GET VNOME
SET DELIMITER TO "[]"       // muda os delimitadores novamente
@ 12,10 SAY "DIGITE O ENDEREÇO..:" GET VENDERECO
READ
```

SET DEVICE

Propósito: Configurar a saída dos comandos @. . . SAY.

Sintaxe: SET DEVICE TO SCREEN|PRINTER

Exemplo:

```
CLEAR
@ 10,10 SAY "LIGUE A IMPRESSORA E PRESS. QQ. TECLA\\"
INKEY(0)        // aguarda qualquer tecla
SET DEVICE TO PRINTER           // liga a saída (@.. say) para a
                                // impressora
@ 20,15 SAY "SERÁ IMPRESSO NA LINHA 20, COLUNA 15 DO PAPEL"
SET DEVICE TO SCREEN            // retorna a saída para a tela
```

SET EPOCH

Propósito: Permite um maior controle das datas que não possuem quatro dígitos no ano.

Sintaxe: SET EPOCH <ano>

Exemplo:

```
SET DATE FORMAT TO "DD/MM/YYYY"      //  formata  o  ano  com  4
dígitos
? CTOD ("04/05/78")         // resultado: 04/05/1978
? CTOD ("04/05/92")         // resultado: 04/05/1992
SET EPOCH TO 1980
? CTOD ("04/05/78")     // resultado: 04/05/2078     //       data
menor?
? CTOD ("04/05/92")         // resultado: 04/05/1992
```

SET ESCAPE

Propósito: Ativar ou desativar a saída de um GET através da tecla <ESC>.

Sintaxe: SET ESCAPE ON|OFF|(.T.)/(.F.)

SET EXACT

Propósito: Determina se as comparações entre expressões caracteres devem ser totalmente iguais ou parciais.

Sintaxe: SET EXACT ON|OFF|(.T.)/(.F.)

Exemplo:

```
//   .T. (sim)        .F.  (não)
SET EXACT OFF              // padrão
? "AB1" = "AB1CD"       // RESULTADO: .T.
? "AB1" = "AB1"            // RESULTADO: .T.
SET EXACT ON
? "AB1" = "AB1CD"       // RESULTADO: .F.
? "AB1" = "AB1"            // RESULTADO: .T.
```

SET EXCLUSIVE

Propósito: Determina se a abertura de arquivos para utilização será de modo exclusivo ou compartilhado.

Sintaxe: SET EXCLUSICE ON|OFF|(.T.)/(.F.)

SET FILTER

Propósito: Cria filtros lógicos que escondem registros que não atendem a condição do filtro criado.

Sintaxe: SET FILTER TO <condição>

Exemplo:

```
USE MALA
SET FILTER TO NOME = "A"          // somente os nomes que começam
                              // com a letra A
LISTA NOME, ENDEREÇO
SET FILTER TO                     // tira o filtro, volta ao normal
LISTA NOME, ENDEREÇO
```

SET FIXED

Propósito: Determina a saída de casas decimais de todos os números.

Sintaxe: SET FIXED ON|OFF(.T.)/(.F.)

SET FORMAT

Propósito: Executa um arquivo de formato de tela quando um READ é avaliado.

Sintaxe: SET FORMAT <rotina>

Exemplo:

```
VNOME:=SPACE(40)
VENDERECO:=SPACE(30)
SET FORMAT TO TELA          // seta o formato para uma procedure
de
                            // nome TELA
READ ──────────────┐
PROCEDURE TELA ◄───┘
@ 10,10 SAY "NOME......:" GET VNOME
@ 12,10 SAY "ENDEREÇO..:" GET VEDERECO
RETURN
```

SET FUNCTION

Propósito: Reprogramar uma tecla de função.

Sintaxe: SET FUNCTION <tecla> TO <expressão caractere>

Exemplo:

```
          // reprogramando as teclas F2 e F3
SET FUNCTION 2 TO "GORKI STARLIN"+CHR(13)      //     CHR(13)     =
<ENTER>
SET FUNCTION 3 TO "EDITORA ERICA"
? "PRESS. <F3> OU <F2>"
ACCEPT "DIGITE ALGO..:" TO TESTE
```

SET INDEX

Propósito: Abrir arquivos de índices para um arquivo de dados aberto na área de trabalho corrente.

Sintaxe: SET INDEX TO <lista de arquivos de índices>

Exemplo:

```
USE MALA
SET INDEX TO INOME, ICEP        // organizado pelo índice NOME
LIST NOME, ENDEREÇO, CIDADE
SET ORDER TO 2       // ICEP, NOME
LIST NOME, ENDEREÇO, CIADE
SET INDEX TO         // fecha todos os índices
```

SET INTENSITY

Propósito: Determina como os campos de edição GET's e PROMPT's serão exibidos.

Sintaxe: SET INTENSITY ON|OFF|(.T.)/(.F.)

SET KEY

Propósito: Determina uma chamada de uma rotina através de uma tecla.

Sintaxe: SET KEY <número da tecla> TO <rotina>

Exemplo:

```
CLEAR
SET KEY -2 TO TERMINA( )        // liga a tecla <f2> com a função
                                // TERMINA( )
VNOME:=SPACE(30)
@ 23,10 SAY "<F2> TERMINA O PROGRAMA"
@ 10,10 SAY "DIGITE O NOME...:" GET VNOME
READ
FUNCTION TERMINA( )
CANCEL
RETURN
```

SET MARGIN

Propósito: Estabelecer o tamanho da margem esquerda para saída para a impressora.

Sintaxe: SET MARGIN TO <tamanho>

Exemplo:

```
USE MALA INDEX INOME
SET MARGIN TO 10
LIST NOME, ENDEREÇO, CIDADE TO RPINTER
```

SET MESSAGE

Propósito: Especifica qual linha do vídeo será utilizada para exibir as mensagens saídas pelo comando Prompt.

Sintaxe: SET MESSAGE TO <linha> (CENTER/CENTRE)

Exemplo:

```
CLEAR
SET MESSAGE TO 23 CENTER
@ 10,10 PROMPT “1 - CADASTRAR“ MESSAGE “CADASTRAMENTO....:”
@ 12,10 PROMPT “2 - PESQUISA “ MESSAGE “PESQUISANDO......:”
MENU TO VAR
      :
      :
      :
```

SET ORDER

Propósito: Estabelecer qual dos arquivos de índices abertos será o Master Index.

Sintaxe: SET ORDER TO <número do índice>.

Exemplo:

```
USE MALA INDEX ICEP, INOME
LIST NOME, ENDEREÇO, CIDADE, CEP// lista em ordem de NOMES
```

```
SET ORDER TO 2              // muda o arquivo de índice de
controle
LIST NOME, ENDEREÇO, CIDADE, CEP// lista em ordem de CEP
```

SET PATH

Propósito: Especificar uma direção de disco ou diretório que será pesquisada pelo Clipper quando este tentar abrir arquivos e não os encontrar.

Sintaxe: SET PATH <lista de direções>

Exemplo:

```
SET PATH TO C:\FOLHA;C:\FATURA        // assinala dois caminhos
                                      // opcionais
```

SET PRINTER

Propósito: Especificar a saída do console para a impressora ou para um arquivo.

Sintaxe: SET PRINTER ON|OFF|(.T.)/(.F.)
SET PRINTER TO <arquivo>
SET PRINTER TO <device>

Exemplo:

```
SET PRINTER OFF
? DATE( ), TIME( )
SET PRINTER ON        // liga a saída da console para impressora
? DATE( ), TIME( )
```

SET PROCEDURE

Propósito: Abrir um arquivo de procedures e compilar suas procedures, colocando-as dentro do programa .OBJ a ser gerado.

Sintaxe: SET PROCEDURE TO <nome do arquivo>

SET RELATION

Propósito: Estabelecer relacionamentos entre áreas de trabalho.

Sintaxe: SET RELATION TO [<campo> | <registro> INTO <área>], TO. . .
[ADDITIVE]

Exemplo:

```
USE CURSOS.DBF INDEX CODCUR.NTX
USE ALUNOS.DBF NEW
SET RELATION INTO CURSO TO CURSOS          //     estabelece     a
relação
```

SET SCOREBOARD

Propósito: Ligar ou desligar a exibição das mensagens emitidas por READ e MEMOEDIT( ).

Sintaxe: SET SCOREBOARD ON|OFF|<.F.>/<.T.>

SET SOFTSEEK

Propósito: Ligar ou desligar a pesquisa relativa do comando SEEK.

Sintaxe: SET SOFTSEEK ON|OFF|(.T.)/(.F.)

SET TYPEAHEAD

Propósito: Determina o tamanho do buffer do teclado.

Sintaxe: SET TYPEAHEAD TO <valor do tamanho>

SET UNIQUE

Propósito: Ligar ou desligar a inclusão de chaves duplicadas em um índice.

Sintaxe: SET UNIQUE ON|OFF|(.T.)/(.F.)

SET WRAP

Propósito: Liga ou desliga a rolagem da barra entre extremos do menu montado pelo comando @. . . PROMPT.

Sintaxe: SET WRAP ON|OFF|(.T.)/(.F.)

SKIP

Propósito: Saltar o ponteiro entre os registros do banco de dados.

Sintaxe: SKIP <salto> [ALIAS <nome da área>]

Exemplo:

```
USE MALA
GO 1
SKIP 2          // salta para o registro 3
SKIP 4          // salta para o registro 7
SKIP -3         // salta para o registro 4
```

SORT

Propósito: Criar um arquivo de dados (.DBF) Classificado.

Sintaxe: SORT TO <arquivo> ON <campo> [/ [A] [D] [C]], <campo2>. . .
[<escopo>] [WHILE <condição>][FOR <condição>]

Exemplo:

```
USE MALA
SORT TO MALA2 ON NOME      //  classificara  os  registros  pelo
campo
                           // NOME
USE MALA2
LIST NOME, ENDEREÇO, CIDADE
```

STATIC

Propósito: Declara uma variável ou matriz como estática.

Sintaxe: STATIC <identificador> [:=<inicializador>]

Exemplo:

```
FUNCTION SENHA
STATIC VCONTROLE := 6      // declara a variável como estática
      :
      :
RETURN
```

STORE

Propósito: Atribuir valores a variáveis.

Sintaxe: STORE <valor> TO <variáveis>

Exemplo:

```
STORE 123.33 TO VAR1       // equivalente a VA1:=123.33
? VAR1                     // mostra o valor de VAR1
VAR1:=VAR2:=4848
? VAR1,VAR2                // resultado: 4848   4848
```

SUM

Propósito: Realizar o somatório de expressões.

Sintaxe: SUM <lista de expressões> TO <lista de variáveis> [<escopo>] [WHILE <condição>] [FOR <condição>]

Exemplo:

```
USE FOLHA
SUM SALÁRIO TO TOTALSAL FOR SETOR = 1 // totaliza o salário
                                      // dos funcionários do setor 1
@ 10,10 SAY "RESULTADO...:"+STR (TOTALSAL)
```

TEXT

Propósito: Permite a exibição de um bloco de textos no vídeo, em um arquivo ou na impressora.

Sintaxe: TEXT [TO PRINTER][TO FILE <arquivo.text> <texto>...
ENDTEXT

Exemplo:

```
TEXT            // abre o bloco de texto
----------------------
ISTO E APENAS UM TEXTO
----------------------
ENDTEXT         // finaliza o bloco de texto
```

TOTAL

Propósito: Cria um arquivo (.DBF), contendo valores totalizados de outros arquivos de dados.

Sintaxe: TOTAL ON <campo> TO <arquivo> [<escopo>] [FIELDS <lista campo> [FOR <condição>]

TYPE

Propósito: Mostrar o conteúdo de um arquivo texto gravado em disco.

Sintaxe: TYPE <arquivo> [TO PRINTER] [TO FILE <arquivo nº2>

Exemplo:

```
TYPE MENU.PRG TO PRINTER        // imprime a listagem do programa
                                // MENU.PRG
```

UNLOCK

Propósito: Liberar travamentos de arquivo ou registro em ambiente de Rede Local.

Sintaxe: UNLOCK[ALL]

Exemplo:

```
USE MALA SHARED
      :
      :
           IF FLOCK( )          // trava todos os registros
              REPLACE SALÁRIO WITH VSAL*INDICE ALL
              UNLOCK            // libera o travamento pendente
           ELSE
              ? "NÃO É POSSÍVEL PROCESSAR OS REGISTROS NO MOMENTO"
           ENDIF
```

UPDATE

Propósito: Atualizar o arquivo aberto na área corrente a partir de outro arquivo de dados aberto em outra área de trabalho.

Sintaxe: UPDATE FROM <área|arquivo> ON <campo chave>
REPLACE <campo> WITH <expressão>,
<campo2> WITH ,<expressão2>,,,
[RANDOM]

USE

Propósito: Abrir um arquivo de dados (.DBF) e opcionalmente arquivo a este associado.

Sintaxe: USE <arquivo.dbf> [index <lista de arquivo de índice>]
[ALIAS <apelido>][EXCLUSIVE/SHARED]
[NEW] [READONLY]
VIA < C driver>

Exemplo:

```
USE MALA INDEX ICOD, INOME
USE MALA READONLY          // somente para leitura
USE FOLHA INDEX CODIFO NEW      //   abre o arquivo na próxima
área
                                // disponível.
```

WAIT

Propósito: Determinar uma pausa na execução do programa até que uma tecla seja pressionada.

Sintaxe: WAIT [<mensagem>] TO [<variável>]

Exemplo:

```
A:=4
WAIT "Press. qualquer tecla para continuar"
B:=5
? A+B
```

ZAP

Propósito: Excluir os registros do arquivo aberto na área corrente.

Sintaxe: ZAP

Exemplo:

```
USE MALA INDEX ICOD, ICEP
ZAP                        // elimina todos os registros.
```

Funções da Linguagem Clipper 5.2

AADD( )

Propósito: Adicionar um novo elemento no final de um valor.

Sintaxe: AADD (<alvo>,<expvalor>)

< alvo > é o vetor no qual será adicionado um novo elemento.

< Expvalor > é o valor a ser atribuído ao novo elemento.

Exemplo:

```
declare vetor [2], vetor2 [2]
vetor [1] = “teste”
vetor [2] = “Gorki”
vetor2 [1] = “Starlin”
vetor2 [2] = “livro”
AADD (vetor, vetor2)       // o AADD( ) adiciona um terceiro
elemento                   // e automaticamente alterando o
tamanho                    // do vetor. O terceiro elemento
será um                    // array bidimencional que tem
como                       // referência o vetor2 [ ].
            //   mostrando os dados dentro dos vetores
? vetor [1]
? vetor [2]
? vetor [3] [1]
? vetor [3] [2]
vetor2 [1] = “última atribuição”
? vetor [3] [1]
? vetor [3] [2]
```

ABS( )

Propósito: Retorna o valor absoluto de uma expressão numérica.

Sintaxe: ABS(<valor numérico>)

<valor numérico> é uma expressão numérica a ser devolvida ao seu valor absoluto.

Exemplo:

```
a : = 9
b : = -4
? abs (a)              // devolve 9
? abs (b)              // devolve 4
```

ACHOICE( )

Propósito: Construir e executar menus do tipo Pop-up.

Sintaxe: ACHOICE(<topo>, <esquerda>, <base>, <direita>, <itens do menu> [<A itens

selecionaveis> <Litensselecionaveis>, <funções do usuário>, <item
inicial>, <linha janela>])

Exemplo:

```
/*    EXEMPLO DE PROGRAMA UTILIZANDO ACHOICE( )
      AUTOR: GORKI STARLIN
*/
CLEAR                              // limpa a tela
LOCAL ITEM  [4], SELEÇÃO [4]
ITEM [1] := "CADASTRAR"            // atribui os valores do vetor
ITEM [2] := "PESQUISAR"
ITEM [3] := "ALTERAR"
ITEM [4] := "EXCLUIR"
SELEÇÃO [1] :=  SELEÇÃO [2] := .T. // determina itens disponíveis
SELEÇÃO [3] :=  SELEÇÃO [4] := .F.              //      determina
itens                                                  //       não
disponíveis
ESCOLHA := ACHOICE (12,12,14,15,ITEM,SELEÇÃO)
? ESCOLHA                                 //   mostra   a   escolha   do
usuário
      DO CASE
            CASE ESCOLHA = 1
                  DO CADASTRA
            CASE ESCOLHA = 2
                  DO PESQUISA
            CASE ESCOLHA = 3
                  DO ALTERA
            CASE ESCOLHA = 4
                  DO ELIMINA
            CASE ESCOLHA = 0
                  CANCEL
      ENDCASE
```

ACLONE( )

Propósito: Duplicar um Array(vetor) do tipo multidimensional.
Sintaxe: ACLONE( )

Exemplo:

```
Local array1, array2
array1  := { 12,13,14 }
array2  := ACLONE (array1)        // array2 é igual a array1,ou
seja
                                  // { 12,13,14 }
```

ACOPY( )

Propósito: Cópia de informações entre vetores.

Sintaxe: ACOPY (<vetor fonte>, <vetor destino>, <início>, <quantos>, < posição destino>).

Exemplos:

```
Local vetor1,vetor2
vetor1 := { 10, 10, 10 }
vetor2     := { 20, 20, 20 }
ACOPY (vetor1, vetor2,1,2}       / / vetor 2 é agora { 10, 10, 20
}
```

ADEL( )

Propósito: Elimina um elemento de um vetor

Sintaxe: ADEL(<vetor>, < posição>)

Exemplo:

```
Private vetor
vetor := {  100,  300,  200  }
ADEL( vetor,2)             // VETOR passa a conter { 100,200,
nil}
```

ADIR( )

Propósito: Armazenar em uma array (VETOR) as informações lidas a partir de um diretório.

Sintaxe: ADIR([<especifica>, <nomes arquivos>, <tamanho>, <datas>, <horas>, <atributos>]).

Exemplo:

```
Private fontes [ADIR (“*.PRG”)]       // cria um vetor com o
                            //tamanho correspondente ao número de
                            // .prg’s existente no diretório
                            // corrente

ADIR (“*.PRG”, FONTES)      // preenche o vetor com o nome dos
                                  // arquivos
ESCOLHA = ACHOICE (10,10,20,35,FONTES)     // monta um menu Pop-
Up
? “SUA ESCOLHA FOI...:“ + STR  (ESCOLHA)
```

AEVAL( )

Propósito: Executar um code block (Bloco de Código) para cada elemento do vetor multidimencional.

Sintaxe: AEVAL (<Vetor>, <Bloco>, [<início>],[<quantidade>])

Exemplo:

```
/*   EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO AEVAL( ) */
LOCAL ARQUIVO : = DIRECTORY (“*.*”), NOMES : = {}
CLEAR
AEVAL (ARQUIVOS {  | FILES | AADD (NOMES, FILES [1]  ) } )
ESCOLHA :=  ACHOICE ( 10, 10, 20, 35, NOMES)
```

AFIELDS( )

Propósito: Preenche os elementos de vetores com a estrutura do banco de dados que estiver aberto na área corrente de trabalho.

Sintaxe: AFIELDS([<campos>], [<tipos>], [<tamanho>], [<decimais>].

AFILL( )

Propósito: Preencher um vetor com um determinado valor.

Sintaxe: AFILL(<vetor destino>, < valor>, <início>, <quantidade>).

Exemplos:

```
Local vetor [5]
Afill (vetor, 4)      // resultado: vetor = {4, 4, 4, 4, 4}
```

AINS( )

Propósito: Inserir um elemento com uma informação NIL (nulo) em um vetor.

Sintaxe: AINS(<vetor>, <posição>)

Exemplo:

```
Private vetor
vetor := {10,  20,  30}
AINS(vetor,2)         // resultado após AINS ( ) ->
                      // vetor  := {10, NIL, 20}
```

ALERT( )

Propósito: Criar uma caixa de diálogo simples com o usuário.

Sintaxe: ALERT (<mensagem string>, <vetor com o opções>)

que o usuário poderá escolher.

Exemplo:

```
Local nEscolha, aOPÇÕES := {"Repetir", "Abortar"}
USE CLIENTES
CLEAR
DO WHILE .NOT. ISPRINTER( )     / / SE NÃO EXISTE IMPRESSORA
            nESCOLHA := ALERT ( "IMPRESSORA NÃO ENCONTRADA";
aOPÇÕES)
            IF nESCOLHA = 2
                RETURN
```

```
            ENDIF
ENDDO
SET PRINT ON          / / LIGA  A  IMPRESSORA
LIST NOME,ENDERECO    // LISTA OS DADOS
SET PRINT OFF         // DESLIGA  A  IMPRESSORA
```

ALIAS( )

Propósito: Retorna o nome do apelido de uma área de trabalho.

Sintaxe: ALIAS (<área de trabalho>)

Exemplo:

```
USE MALA NEW
ÁREA = SELECT( )
USE CLIENTE NEW
? ALIAS (ÁREA)
```

ALLTRIM( )

Propósito: Remover todos os espaços em branco que existirem em uma cadeia de caracteres.

Sintaxe: ALLTRIM (<cadeia caracteres>)

Exemplo:

```
PRIVATE STRING
STRING : = SPACE(30)+ " GORKI " + " STARLIM "
? STRING
? ALLTRIM (STRING)
```

ALTD( )

Propósito: Ativar o Clipper Debugger.

Sintaxe: ALTD (ação)

Exemplo:

```
/*    FOLHA.PRG
      AUTOR: GORKI STARLIN
*/
```

```
PARAMETER AÇÃO          // recebe o parâmetro enviado a partir do
                        // sistema operacional
      IF AÇÃO           // verifica se o parâmetro é nulo
            AÇÃO = 0          // atribui 0 a variável ação
      ELSE
            AÇÃO = VAL (AÇÃO)
      ENDIF
            ALTD (AÇÃO)           // invoca a função e configura e
                                  // Debugger
:
:
<declarações>
:
:
```

ARRAY( )

Propósito: Cria um array de tamanho especificado e sem inicialização.

Sintaxe: ARRAY (<elementos> [,elementos..]).

Exemplo:

```
MATRIZ := ARRAY (2,3)      //    cria    definindo    apenas    as
dimensões..
MATRIZ := {{4,3,4}, {6,3,2}}     // cria atribuindo valores.
```

ASC( )

Propósito: Devolve o código ASCII (0 a 255) de um determinado caractere.

Sintaxe: ASC (<caractere(s)>)

Exemplo:

```
? ASC (“a”)      // resposta: 97
? ASC (“A”)      // resposta: 65
? ASC (“ ”)           // resposta: 0 (nulo)
```

ASCAN( )

Propósito: Pesquisar em um vetor uma informação ou bloco de código (code block).

Sintaxe: ASCAN (<vetor>,<procurar>, [<início>],[<quantidade>].

Exemplo:

```
VETOR := {“BATATA”, “TOMATE”, “FEIJÃO”, “CARNE” }
? ASCAN (VETOR, “BATATA”}        // resultado: 1
ENDEREÇO := ASCAN (VETOR, “FEIJÃO”)         //resultado:
endereco=3
? endereço
? ascan (vetor, {|var| upper (var) == “TOMATE”}) //code block
```

ASIKE( )

Propósito: Alterar o número de elementos de um vetor.

Sintaxe: ASIZE (<vetor>, <tamanho>)

Exemplo:

```
VETOR := {“BATATA”, “TOMATE”, “FEIJÃO”, “CARNE”}
? LEN (VETOR)         // mostra o tamanho do vetor. resultado: 4
ASIZE (VETOR,10)// altera o tamanho do vetor
? LEN (VETOR)         // resultado: 10
  FOR I = 1 TO LEN (VETOR)       // mostra todos os elementos do
                                 // vetor
     ? VETOR [i]
  NEXT
```

ASORT( )

Propósito: Coloca em ordem os elementos de um vetor.

Sintaxe: ASORT (<vetor>,[<início>],[<quantidade>],[<ordem>]

Exemplo:

```
VETOR : {“BATATA”, “TOMATE”, “FEIJÃO”, “CARNE” }    // cria o
```

```
                                                        // vetor
ASORT (VETOR)              // ordem ascendente
   FOR I = 1 TO LEN (VETOR)// mostra todos os elementos do vetor
     ? VETOR [i]
   NEXT
ASORT (VETOR,,, {|a, b| a > b })      // ordem descendente
   FOR i = 1 TO LEN (VETOR)// mostra todos os elemento do vetor
     ? vetor [i]
   NEXT
```

AT( )

Propósito: Mostra o endereço de uma string dentro de um cadeia de caracteres.

Sintaxe: AT (<string>,<cadeia>).

Exemplo:

```
VAR := "BATATA"
? AT ("TA", VAR)           // resultado: 3
? AT ("T", VAR)            // resultado: 2
? AT ("Z", VAR)            // resultado: 0
```

ATAIL( )

Propósito: Retornar o valor do último elemento do vetor.

Sintaxe: ATAIL (<vetor>).

Exemplo:

```
LOCAL aNOMES := {"MARIA", "JOSÉ", "JOÃO", "ANA"}
ULTIMO := ATAIL (aNOMES)
? ÚLTIMO        // ANA
```

BIN2( )

Propósito: Realizar a conversão de um valor inteiro de 16 bits para um valor numérico.

Sintaxe: BIN2 (<cadeia>).

BIN2L( )

Propósito: Realiza a conversão de um valor inteiro de 32 bits para um valor numérico.

Sintaxe: BIN2L (<cadeia>).

BIN2W( )

Propósito: Realiza a conversão de um valor inteiro sem sinal 16 bits para um valor numérico.

Sintaxe: BIN2W (<cadeia>)

BOF( )

Propósito: Retornar se o posicionamento interno de um banco de dados encontra-se no início do arquivo (Begin of File).

Sintaxe: BOF( )

Exemplo:

```
USE MALA NEW                // abre o arquivo de dados
? BOF( )                    // resultado: .F.
SKIP - 1                    // pule - 1
? BOF( )                    // resultado: .T.
```

BROWSE( )

Propósito: Folhear um banco de dados dentro de uma janela.

Sintaxe: BROWSE(<linha_inicial>, <coluna_inicial>, <linha_ final>, <coluna_final>).

Exemplo:

```
USE MALA NEW          // abre o banco de dados
L_INICIAL = 5         // cria variáveis p/ coordenadas da janela
C_INICIAL = 5
F_FINAL   = 22
C_FINAL   = 67
                      // desenha uma moldura
@ L_INICIAL-1,C_INICIAL-1 TO L_FINAL+1,C_FINAL+1 DOUBLE
                      // folheia o b.d.
BRAWSE(L_INICIAL, C_INICIAL, L_FINAL, C_FINAL)
```

CDOW( )

Propósito: Extrair de uma data de uma expressão caracteres referente ao dia da semana da data.

Sintaxe: CDOW(<data>).

Exemplo:

```
? DATE                // mostra a data do sistema
? CDOW(DATE( ))       // mostra o dia da semana da data do sistema
? CDOW(DATE( )+10)    // mostra o dia da semana de dez dias após a
                      // data do sistema.
```

CMONTH( )

Propósito: Analisar uma data e devolver o nome do mês correspondente.

Sintaxe: CMONTH(<data>).

Exemplo:

```
// Os exemplos a seguir, demostram a utilização da função
// CMOUNTH( ).
? CMOUNTH(DATE( ))         // resultado: mês da data do sistema
? CMOUNTH(DATE( )+30)      // resultado: mês posterior à data do
                           // sistema.
```

COL( )

Propósito: Devolver a coordenada atual Cursor em tela referente à posição da coluna.

Sintaxe: COL( ).

Exemplo:

```
CLEAR                  // limpa a tela
LOCAL VNOME := "JOÃO", VSALARIO:=39000.00
@ 05,10 SAY "NOME.....:" + VNOME
@ 07,10 SAY "SALÁRIO..:"
@ 07, COL( ) +2 SAY SALÁRIO
```

COLORSELEC( )

Propósito: Ativar um atributo na configuração de cores corrente.

Sintaxe: COLORSELECT(nCOR).

Exemplo:

```
      SETCOLOR("B/W","N/W","GR/W","N/GR")
? "GORKI"
COLORSELECT( 1 )
? "GORKI"
COLORSELECT( 0 )
? "GORKI"
```

CTOD( )

Propósito: Transformar uma expressão caractere em uma data.

Sintaxe: CTOD(<expressão>).

Exemplo:

```
PRIVATE CAR, VARDATA
CAR := "20/1/93"
?CTOD(CAR) + 365      // mostra 365 dias após o conteúdo da
```

```
                          // expressão caractere contida em CAR.
VARDATA:= CTOD(“  /  /  “)  // cria uma variável do tipo data em
                          // branco.
```

CURDIR( )

Propósito: Mostra o nome do diretório atual de uma determinada unidade de disco.

Sintaxe: CURDIR( ).

Exemplo:

```
? CURDIR( )
```

DATE( )

Propósto: Retornar a data do sistema operacional.

Sintaxe: DATE( ).

Exemplo:

```
? DATE( )             // mostra a data do sistema
VARDATA := DATE( )    // cria uma variável contendo a data do
                      // sistema, sendo que o tipo da variável
                      // será D.
? DATE( ) + 4         // mostra a data do sistema + 4 dias
SET DATE ITAL
? DATE( )
```

DAY( )

Propósito: Mostra um número correspondente ao dia de uma data.

Sintaxe: DAY( ).

Exemplo:

```
? DATE( )             // mostra a data do sistema operacional
? DAY(DATE( ))        // mostra o dia da data do sistema
                      // operacional.
```

DBAPPEND( )

Propósito: Criar (inserir) um registro em branco no banco de dados aberto na área corrente de trabalho.

Sintaxe: DBAPPEND( )

Exemplo:

```
/*
      NOME DO PROGRAMA: CADMULT1.PRG
           AUTOR : GORKI STARLIN
           FUNÇÃO: ESTE MODULO ANEXA DADOS NO ARQUIVO PAGAMENTO
  CARACTERÍSTICA: REDE LOCAL
*/
LOCAL CODVAR, NOMEVAR, SETORVAR, CARGOVAR, ATIVOVAR, DATAVAR
USE COLHA INDEX CODX,NOMEX
IF NETERR( )        // testa se houve erro na abertura do arquivo
   ? "O arquivo de dados não se encontra disponível"
 INKEY(0)
   CANCEL
ENDIF
DO WHILE .T.
                // lay out
CLEAR
      SET COLOR TO W+/N
      SET COLOR TO
      @@ 01,01 TO 24,79 DOUBLE
      @@ 02,02 TO 04,78
      @@ 03,03 SAY "SÍRIOS INFORMÁTICA"
      @@ 03,60 SAY DATE( )
      @@ 03,70 SAY TIME( )
           // cria variáveis
CODVAR     = 0
SETORVAR   = 0
SALARIOVAR = 0
NOMEVAR    = SPACE(35)
CARGOVAR   = SPACE(15)
ATIVOVAR   = (.T.)
DATAVAR    = CTOD("  /  /  ")
           // entrada de dados
      @@ 06,10 SAY "** CADASTRAMENTO DE FUNCIONÁRIOS **"
      @@ 08,10 SAY "CÓDIGO.......:" GET CODVAR PICTURE "9999"
      READ
      IF CODVAR = 0   // verifica se o usuário não digitou o
                      // código
         OP:="S"      // cria a variável OP
           @@ 21,15 SAY "SAI DESTE MODULO.(S/N).:" GET OP PICT "A"
```

```
         READ
         IF OP = "S" // verifica a resposta do usuário
              RETURN     // retorne
         ENDIF
       LOOP          // sobe a execução p/ linha do DO WHILE
    ENDIF            // fim do se
       SEEK CODVAR        // pesquisa no índice o conteúdo da
                          // variável CODVAR
  IF EOF( )
     DBAPPEND( )         // tenta criar um registro em branco
     DO WHILE NETRR( )   // faça enquanto HOUVER ERRO
        DBAPPEND( )      // tenta (novamente criar o registro
     ENDDO               // fim do faça enquanto
              // entra com o restante dos dados do funcionário
    @@ 10,10 SAY "NOME DO FUNCIONÁRIO..:" GET NOMEVAR PICT "@!"
    @@ 12,10 SAY "SETOR TRABALHO.......:" GET SETORVAR PICT "9"
    @@ 14,10 SAY "CARGO FUNCIONAL......:" GET CARGOVAR PICT "@!"
    @@ 16,10 SAY "SALÁRIO.......:" GET SALÁRIO PICT "9999999.99"
    @@ 18,10 SAY "FUNCIONÁRIO ATIVO....:" GET ATIVOVAR
    @@ 20,10 SAY "DATA ADMISSÃO........:" GET DATAVAR
    READ
              // grava os dados no registro em branco
         REPLACE NOME WITH NOMEVAR
         REPLACE SETOR WITH SETORVAR
         REPLACE CARGO WITH CARGOVAR
         REPLACE ATIVO WITH ATIVOVAR
         REPLACE DTADM WITH DATAVAR
         REPLACE SALÁRIO WITH SALARIOVAR
         @@ 21,20 SAY "** CADASTRADO **"
         WAIT " "            // aguarda qq tecla
         COMMIT              // atualiza fisicamente o registro
         REPLACE COD WITH CODVAR
         UNLOCK              // libera o registro criado
  ELSE       // se não
         @@ 21,20 SAY "** REGISTRO JÁ CADASTRADO **"
         WAIT " "            // aguarda qq tecla
  ENDIF
ENDDO
```

DBCLEARFIL( )

Propósito: Limpar a condição de filtro ativo.

Sintaxe: DBCLEARFIL( )

Exemplo:

```
USE CLIENTES
SET FILTER TO NOME = "A"          // separa os registros
BROWSE( )
DBCLEARFILTER( )           // equivalente : SET FILTER TO
```

DBCLEARINDEX( )

Propósito: Desativar todos os índices abertos para um arquivo de dados.

Sintaxe: DBCLEARINDEX( ).

DBCLEARRELATION( )

Propósito: Desativar o relacionamento entre arquivos.

Sintaxe: DBCLEARELATION( ).

DBCLOSEALL( )

Propósito: Fechar todos os arquivos de dados.

Sintaxe: DBCLOSEALL( ).

DBCOMMIT( )

Propósito: Atualizar fisicamente no arquivo em disco, alterações que estão no buffer.

Sintaxe: DBCOMMIT( ).

Exemplo:

```
:
:
:
 REPLACE NOME WITH NOMEVAR
 REPLACE SETOR WITH SETORVAR
 REPLACE CARGO WITH CARGOVAR
 REPLACE ATIVO WITH ATIVOVAR
 REPLACE DTADM WITH DATAVAR
 REPLACE SALÁRIO WITH SALARIOVAR
 @@ 21,20 SAY "** CADASTRADO **"
 INKEY(0)       // aguarda qq tecla
 DBCOMMIT( )          // atualiza o arquivo fisicamente.
```

DBCOMMITALL( )

Propósito: Atualizar fisicamente todos os arquivos abertos, suas alterações que estão no buffer.

Sintaxe: DBCOMMIT( ).

DBCREATE( )

Propósito: Criar um banco de dados (.DBF) a partir de uma estrutura de um arquivo DBF armazenado em um vetor.

Sintaxe: DBCREATE (<arquivo>, <vetor>).

Exemplo:

```
/*    PROGRAMA: CRIA.PRG
      AUTOR: GORKI STARLIN
*/
IF .NOT. FILE (“FUNC.DBF”)       // se func.dfb não existe
      ? “CRIANDO BASE DE DADOS”        // aviso ao operador
      ESTRU:={}        && CRIA UMA MATRIZ
      AADD(ESTRU, {“COD”,”N”,4,0}) // crias os subvetores com
      AADD(ESTRU, {“NOME”,”C”,30,0})    // os campos
      AADD(ESTRU, {“SETOR”,”N”,1,0})
      AADD(ESTRU, {“CARGO”,”C”,15,0})
      AADD(ESTRU, {“SALARIO”,”N”,10,2})
      AADD(ESTRU, {“DTADM”,”D”,8,0})
      AADD(ESTRU, {“OBS”,”C”,10,0})
      DBCREATE(“FUNC”,ESTRU)      // cria o B.D. (func.dbf) a
                                        // partir da matriz
ENDIF           // fim do se
   :
   :
   :
```

DBCREATEINDEX( )

Propósito: Criar um arquivo de índice para um determinado banco de dados em uso.

Sintaxe: DBCREATEINDEX(<nome index> , <cheve index>, <bloco>, <Lunico>).

Exemplo:

```
LOCAL VNOME
USE CLIENTES
CLEAR
? "INDEXANDO"
DBCREATEINDEX ("NOME", "NOME", {|| NOME})
     // indexa o arquivo pelo nome e cria o arquivo que conterá o
     // controle de índice NOME.NTX
LOCAL VNOME:= SPACE(30)
@ 10,10 SAY "DIGITE O NOME..:" GET VNOME PICTURE "@!"
READ
? "PESQUISANDO"
SEEK VNOME
IF FOUND( )          // se existir
     DISPLAY NOME, ENDEREÇO, CIDADE       // mostra o registro
ELSE
? "REGISTRO NÃO ENCONTRADO"
ENDIF
```

DBDELETE( )

Propósito: Marcar um registro para ser apagado.

Sintaxe: DBDELETE( ).

Exemplo:

```
USE FOLHA INDEX INOME.NTX
SEEK "JOÃO"
IF FOUND( )
 DBDELETE( )         // marca o registro encontrado
ENDIF
DISPLAY ALL NOME, SALÁRIO, COD       // mostra os registros
SET DELETE ON        // filtra os registros marcados
DISPLAY ALL NOME, SALÁRIO, COD       // mostra os registros
RECALL ALL           // recupera todos os registros
DISPLAY ALL NOME, SALÁRIO, COD       // mostra os registros
```

DBEDIT( )

Propósito: Folheia os registros d um banco de dados em uma janela.

Sintaxe: DBEDIT ([<linha_inicial>, <coluna_inicial>, <linha_final>, <coluna_final>, (<vetor de colunas>, "<função do usuário>", < vetor de máscaras>, <máscara>, <vetor de cabeçalhos>, <cabeçalho>, <vetor separador cabeçalhos>, <separador cabeçalho>, <vetor separador de rodapé>, <separador de rodapé>, <vetor rodapé das colunas>, <rodapé das colunas>]).

Exemplo:

```
/*    ESTE PROGRAMA É UM EXEMPLO DA FUNÇÃO DBEDIT
      AUTOR : GORKI STARLIN
/*

USE FOLHA    // abre os arquivos folha.dbf
DECLARE VECTOR_CAMPOS[7]   // declara o vetor que representará os
                           // campos do arquivo a ser editado
           // ARMAZENA OS CAMPOS DO ARQUIVO NOS VETORES
VETOR_CAMPOS [1] = "COD"
VETOR_CAMPOS [2] = "NOME"
VETOR_CAMPOS [3] = "SETOR"
VETOR_CAMPOS [4] = "SALÁRIO"
VETOR_CAMPOS [5] = "CARGO"
VETOR_CAMPOS [6] = "ATIVO"
VETOR_CAMPOS [7] = "DTADM"
           // CRIA VARIÁVEIS P/ DEFINIR A ÁREA DE EDIÇÃO DOS
DADOS
L_INICIAL = 5
C_INICIAL = 5
L_FINAL = 22
C_FINAL = 67
```

```
@ L_INICIAL-1, C_INICIAL-1 TO L_FINAL+1, C_FINAL+1 DOUDLE
DBEDIT  (L_INICIAL,  C_INICIAL,  L_FINAL,  C_FINAL,  VETOR_CAMPOS,
"EDITA")
FUNCTION EDITA ( MODO, ÍNDICE )
SET COLOR TO W+/N
TECLA = LASTKEY ( )
CAMPO = VETOR_CAMPOS [ÍNDICE]
RETORNA = 1
     IF MODO = 4
          IF TECLA = 27
               RETORNA = 0
          ELSEIF TECLA = 13
               @  ROW(),COL() GET & CAMPO
               READ
          ENDIF
     ENDIF
SET COLOR TO
RETURN RETORNA
```

DBEVAL( )

Propósito: Executa e avalia um bloco de código (code block) para cada registro que atenda uma condição ou Escopo.

Sintaxe: DBEVAL(<Bloco>, [<Condição1>, <Condição2>, <Quantidade> <Registro>, <Restante>]).

Exemplo:

```
/*   REAJUSTE DE SALÁRIO
     AUTOR: GORKI STARLIN
*/
LOCAL ÍNDICE := 20, SETVAR := 1
          // cria o bloco de código executável
BLOCO := {|| SALÁRIO := SALÁRIO * ÍNDICE/100 + SALÁRIO}
          // cria o bloco de código contendo a condição FOR
CONDIÇÃO1 := {|| SETOR = SETVAR}
DBEVAL (BLOCO, CONDIÇÃO1)       // mostra e avalia os blocos de
                                // códigos
? "SALÁRIO ATUALIZADOS !"
QUIT
```

DBF( )

Propósito: Retornar o ALIAS (apelido) do banco de dados aberto na área de trabalho corrente.

Sintaxe: DBF( ).

Exemplo:

```
USE FOLHA NEW    // abre o arquivo de dados na próxima área de
                 // trabalho disponível
NOME := DBF( )   // armazena o nome do banco de dados na variável
? NOME           // mostra o conteúdo do variável.
```

DBFILTER( )

Propósito: Devolver uma cadeia de caracteres referente ao filtro estabelecido por SET FILTER.

Sintaxe: DBFILTER( ).

Exemplo:

```
USE FOLHA NEW          // abre o banco de dados
SET FILTER TO SALÁRIO < 40000.00// estabelece um filtro aos
                       // registros a serem processados
LIST NOME, CARGO, SETOR, SALÁRIO
?DBFILTER( )           // resultado: SALÁRIO < 40000.00
```

DBGOBOTTOM( )

Propósito: Desloca o ponteiro interno do arquivo de dados para o último registro lógico do banco de dados.

Sintaxe: DBGOBOTTOM( ).

Exemplo:

```
USE FOLHA
DBGOBOTTOM( )          // vá para o fim do arquivo
DISPLAY NOME, COD, SALÁRIO
```

DBGOTO( )

Propósito: Deslocar o ponteiro interno do arquivo de dados para um determinado registro lógico.

Sintaxe: DBGOTO(<nregistro>).

Exemplo:

```
USE FOLHA
DBGOTO( 6 )     // vá para o registro (record) número 6
DISPLAY NOME, COD, SALÁRIO
```

DBGOTOP( )

Propósito: Deslocar o ponteiro interno do arquivo para o primeiro registro do mesmo.

Sintaxe: DBGOTOP( ).

Exemplo:

```
USE FOLHA
DBGOTOP( )          // vá para o início do arquivo
DISPLAY NOME, COD, SALÁRIO
```

DBRECALL( )

Propósito: Recuperar (desmarcar) registro marcados no arquivo de dados.

Sintaxe: DBRECALL( ).

Exemplo:

```
USE FOLHA INDEX INOME.NTX
SEEK "JOÃO"
IF DELETED( )         // se estiver marcado (deleted)
     DBRECALL( )
     ? "REGISTRO RECUPERADO"
ENDIF
```

DBREINDEX( )

Propósito: Recriar os índices ativos no arquivo de dados.

Sintaxe: DBREINDEX( ).

DBRELATION( )

Propósito: Devolver uma cadeia de caracteres que descreve a expressão usada para estabelecer o relacionamento de dados entre banco de dados através do comando SET RELATION.

Sintaxe: DBRELATION(<relacionamento>).

Exemplo:

```
USE FACULD NEW              // abre o arquivo de dados
USE CURSOS NEW              // abre o arquivo de dados
USE ALUNOS NEW              // abre o arquivo de dados
SET RELATION TO CODCUR INTO CURSOS, CODFACUL INTO FACULDADE
? DBRELATION(2)             // resultado: CODFACUL
? DBRSELECT( )              // resultado: 3
```

DBRSELECT( )

Propósito: Devolver número da área de trabalho a que se destina um relacionamento.

Sintaxe: DBSELECT(<relacionamento>).

Exemplo:

```
USE FACULDAD NEW      // abre o arquivo de dados
USE CURSOS NEW              // abre o arquivo de dados
USE ALUNOS NEW              // abre o arquivo de dados
                // monta dois relacionamentos
SET RELATION TO CODCUR INTO CURSOS, CODFACUL INTO FACULDADE
? DBRELATION(2)             // resultado: CODFACUL
? DBRSELECT(2)              // resultado: 3
? ALIAS(DBRSELECT(2))       // resultado: faculdade
```

DBSEEK( )

Propósito: Pesquisarmos registro do banco de dados indexado numa chave especificada.

Sintaxe: DBSEEK <chave>,[.T./.F.].

Exemplo:

```
USE MALA INDEX INOME
USE CLIENTES INDEX IESTADO NEW
MALA → (DBSEEK("JOÃO", .F.))     // pesquisa no mala o nome JOÃO
                                 // SER SEFTSEEK OFF
     IF FOUND( )           // se existir
          DISPLAY NOME, ENDEREÇO, CIDADE
     ELSE
          ? "NÃO ENCONTRADO !"
          ? RECNO( )       // EOF( )
     ENDIF
```

DBSELECTAREA( )

Propósito: Seleciona uma área de trabalho.

Sintaxe: DBELECTAREA(<área> | <apelido>).

Exemplo:

```
USE MALA INDEX INOME
     DBSELECTAREA( 0 )     // seleciona a próxima área disponível
USE FOLHA INDEX CODF
LIST NOME, SALARIO,SETOR, COD
     DBSELECTAREA(MALA)    // seleciona o arquivo área MALA
     LIST COD, CLIENTE, CIDADE
LIST MALA→CLIENTE, FOLHA→SALÁRIO     // lista registro de outra
                                     // área
MALA→ (DBAPPEND( ))        // cria um registro em branco no
                           // arquivo mala.dbf
```

DBSETDRIVER( )

Propósito: Retornar o nome do driver de arquivo em uso, ou ainda trocar o tipo do driver de arquivo em uso.

Sintaxe: DBSETCRIVER(<NOME DRIVER>).

Exemplo:

```
       :
       :
DBSETDRIVER (“DBFNDX”)
     IF (DBSETDRIVER <> “DBFNDX”)
          ? “O DRIVER .NDX NÃO VÁLIDO”
     ENDIF
       :
       :
```

DBSETINDEX( )

Propósito: Abrir um arquivo de índice em uma área de trabalho.

Sintaxe: DBSETINDEX(<nome do índice>).

Exemplo:

```
USE FOLHA
DBSETINDEX(“INOME”)
DBSETINDEX(“ISOBRENO”)
IF FOLHA-> (DBSEEK (“SILVA”))
     ? FOLHA-> NOME, FOLHA->INDEXRECO
ELSE
     ? “REGISTRO NÃO ENCONTRADO”
ENDIF
```

DBSETORDER( )

Propósito: Ativar um determinado índice aberto como índice mestre do banco de dados.

Sintaxe: DBSETORDER(<Número do índice>).

Exemplo:

```
USE FOLHA
SET INDEX TO INOME, ISOBRENOME
     :
     :
     :
DBSETORDER(2)        // seta o segundo índice aberto como
                     // principal, isto é ISOBRENOME
DBSEEK ("SILVA")
```

DBSETRELATION( )

Propósito: Relacionar duas área (arquivos) de trabalho.

Sintaxe: DBSETRELATION(<Narea>| <apelido>, <bloco expressão> [<Cexpr>]).

Exemplo:

```
USE CLIENTES INDEX ICODCLI
USE VENDAS NEW
DBSETRELATION("CLIENTES", {|| VENDAS->CODVENCLI},;
              "VENDAS->CODVENCLI")
LIST CLIENTES->NOME, VENDAS->VALOR
```

DBSKIP( )

Propósito: Saltar o ponteiro entre os registros do banco de dados.

Sintaxe: DBSKIP (<valor do salto>).

Exemplo:

```
USE FOLHA
GO 1
DISPLAY
DBSKIP( 4 )          // resultado:   RECNO( ) = 5
DISPLAY
SKIP - 2             // resultado:   RECNO( ) = 3
DISPLAY
```

DBSTRUCT ()

Propósito: Criar uma matriz com duas dimensões contendo a estrutura de um banco de dados.

Sintaxe: DBSTRUCT ( ).

Exemplo:

```
# INCLUDE "DBSTRUCT.CH"
LOCAL ESTRUTURA
USE FOLHA NEW              // abre o banco de dados
ESTRUTURA := DBSTRUCT( )        // armazena a estrutura do banco
                                // de dados em ESTRUTURA
                      // usa bloco de código p/ existir dados da
                      // estrutura
AEVAL( ESTRUTURA, {|CAMPO| QOUT (CAMPO [DBS_NAME])})
```

DBUNLOCKALL ( )

Propósito: Liberar todos os travamentos sobre as áreas de trabalho.

Sintaxe: DBUNLOCKALL ( ).

Exemplo:

```
USE FOLHA SHARED NEW
USE CLIENTES SHARED NEW
FOLHA → (FLOCK())
CLIENTES → (FLOCK())      // trava o arquivo
DBUNLOCKALL( )       // libera todos os travamentos
```

DBUSEAREA ( )

Propósito: Abrir um arquivo em uma área de trabalho.

Sintaxe: DBUSEAREA (<Lnome área>,<nome driver>, <arquivo> <apelido>,<Lconpartilhado>,<Lapenas leitura>).

Exemplo:

```
DBUSEAREA (.T., "DBFNTX","VENDAS")     // abre o arquivo vendas
BROWSE( )
```

DELETED ( )

Propósito: Verificar se o registro corrente se encontra deletado (marcado) através do comando DELETE.

Sintaxe: DELETED ( ).

Exemplo:

```
USE FOLHA NEW          // abre os arquivo de dados
USE CARGOS NEW
DISPLAY ALL FR DELETED( )          // mostra todos os registros
                                   // Deletados
DISPLAY ALL FOR FOLHA→(DELETED( ))
```

DESCEND ( )

Propósito: Criar chaves de índices em ordem descendente.

Sintaxe: DESCEND ( ).

Exemplo:

```
USE FOLHA NEW                 // abre o arquivo de dados
INDEX ON DESCEND (nome) TO NID.NTX     // cria a chave de índice
LIST NOME, COD, SALÁRIO, CARGO
     // para utilizar o comando SEEK para fins de pesquisa não se
     // esqueça de declarar DESCEND( ).
```

DEVPOS ( )

Propósito: Movimentar a cabeça de impressão para uma nova posição especificada.

Sintaxe: DEVPOS (<linha>,<coluna>).

Exemplo:

```
SET DEVICE TO PRINT
@ 01,01 SAY "EXEMPLO"
DEVPOS(15,20)          // desloca a cabeça de impressão exibe na
```

```
                    // linha 15, coluna 20
@ PROW( ), PCOL( ) SAY "AS COORDENADAS MUDARAM"
```

DEVOUTPICT ( )

Propósito: Mostra uma informação de qualquer ponto da tela, como característica de informação de saída.

Sintaxe: DEVOUTPICT (<informação> , <cláusula picture> , [<cor>]).

Exemplo:

```
DEVPOS (5,5)
DEVOUT("GORKI STARLIN", "@!", "B/W")
```

DIRECTORY ( )

Propósito: Criar uma matriz multidimensional e armazenar nesta, informações sobre um diretório.

Sintaxe: DIRECTORY (<diretório>, <atributos>).

Exemplo:

```
# INCLUDE "DIRECTRY.CH"          // inclui o arquivo de definições
LOCAL DIRETÓRIO := DIRECTRY ("*.DBF", "D")        //     lê     o
diretório
                    // avalia e executa o bloco de código
AEVAL ( diretório, {|arquivo| qout (arquivo [F_NAME])})
```

DISKSPACE ( )

Propósito: Retornar o espaço livre (em Bytes) de uma determinada unidade de disco.

Sintaxe: DISKSPACE (<drive>).

Exemplo:

```
FUNCTION COPIASEG( )
```

```
          // calcula o tamanho da cópia
TAMANHO :=INT((RECSIZE * LASTREC + HEADER + 1 ))
          // verifica se é possível a cópia no diskete
IF DISKSPACE(1) < tamanho // drive A
     RETURN .F.      // retorne não possível
ELSE
     COPY TO A:BACKUP.DBF     // gera uma copia do b.d com o
nome
                     // backup.dbf no diskete da drive A
     RETURN .T.      // retorne cópia OK!.
```

DOSERROR( )

Propósito: Devolver o código do último erro processado pelo D.O.S.

Sintaxe: DOSERROR( ).

DOW( )

Propósito: Extrair de uma data um número que especifica o dia da semana da mesma.

Sintaxe: DOW(<data>).

Exemplo:

```
// mostra o dia da semana da data de hoje (sistema)
? DOW (DATE( ))      // na forma de número
? DOW (DATE( ))      // na forma de uma cadeia de caracteres
DIA := 1
     DO WHILE DIA <= 7          // faça enquanto dia <= 7
          ? DOW(DIA), CDOW(DIA)      // mostra o dia da semana
          DIA ++          // equivalente a dia:= dia+1
     ENDDO
```

DTOC( )

Propósito: Converter um valor data para uma expressão caractere.

Sintaxe: DTOC(<data>).

Exemplo:

```
? DATE( )            mostra a data de hoje (sistema)
? "DATA DE HOJE É..:" + DTOC(DATE( ))      // mostra a data, no
     // formato de expressão caractere, concatenado em conjunto
     // com uma cadeia de caracteres.
```

EMPTY( )

Propósito: Verifica se uma expressão é vazia.

Sintaxe: EMPTY(<expressão>).

Exemplo:

```
 VCOD := SPACE(5)
@ 10,10 SAY "DIGITE O CÓDIGO....:" GET VCOD PCT "99999";
VALID .NOT. EMPTY(VCOD)
READ
```

EOF( )

Propósito: Verificar se o ponteiro lógico de registros se encontra no fim do arquivo.

Sintaxe: EOF( ).

Exemplo:

```
USE MALA INDEX CODI
VCOD:= SPACE(5)
@ 10,20 SAY "CÓDIGO DO FUNCIONÁRIO A PESQUISAR..:" GT VCOD
READ
SEEK VCOD       // pesquisa o código digitado
     IF EOF( )       // se for o final do arquivo
          ? "registro não encontrado"
     ELSE
          DISPLAY NOME, ENDEREÇO
     ENDIF
```

ERRORBLOCK( )

Propósito: Avaliar um bloco de código (Code Block) quando é detectado um erro no programa em tempo de execução.

Sintaxe: ERRORBLOCK(<errorhandler>).

ERRORLEVEL( )

Propósito: Retornar ou configurar o código de retorno de error do Clipper.

Sintaxe: ERRORLEVEL(código de retorno).

EVAL( )

Propósito: Executar um Bloco de Código.

Sintaxe: EVAL(<bloco>, <lista de argumentos>).

Exemplo:

```
BLOCO := {| ARGUMENTO| ARGUMENTO + 1}
? EVAL (BLOCO,4)     // resultado: 5
```

EXP( )

Propósito: Calcular o E ** X.

Sintaxe: EXP(<expoente>).

Exemplo:

```
? EXP(1)        // resultado: 2.72
```

FCLOSE( )

Propósito: Fechar um arquivo do tipo binário aberto.

Sintaxe: FCLOSE(<número>).

Exemplo:

```
HANDLE:= FOPEN ("LEIAME.TXT")    // abrindo e obtendo o HENDLE
      :
      :
      :
      :
IF FCLOSE(HANDLE)     // fecha o arquivo LEIAME.TXT
     ? "ARQUIVO FECHADO"
ELSE       // senão conseguir fechar o arquivo
     ? "NÃO FOI POSSÍVEL FECHAR O ARQUIVO"
ENDIF
```

FCOUNT( )

Propósito: Retornar a quantidade de campos do arquivo de dados (.DBF) aberto na área corrente de trabalho.

Sintaxe: FCOUNT( ).

Exemplo:

```
USE MALA
USE CADASTRO
? FCOUNT( )                 // resultado: 7
? MALA → (FCOUNT( ))        // resultado: 5
```

FCREATE( )

Propósito: Criar um arquivo binário de tamanho zero.

Sintaxe: FCREATE(<arquivo>, [<atributo>]).

Exemplo:

```
HANDLE := FCREATE ("LEIAME.TXT",0)          // cria o arquivo
     IF HANDLE = -1
          ? "houve erro na criação do arquivo"
     ELSE
          FWRITE (HANDLE, "ISTO E UMA MENSAGEM")
          FCLOSE (HANDLE)
ENDIF
```

FERASE( )

Propósito: Apagar arquivo do disco.

Sintaxe: FERASE (<arquivo>).

Exemplo:

```
OPERAÇÃO := FERASE("LEIAME.TXT")
     IF OPERAÇÃO = -1
          ? "ERRO !, ARQUIVO NÃO FOI APAGADO"
     ELSE
          ? "ARQUIVO APAGADO"
     ENDIF
```

FERROR( )

Propósito: Analisar se houve erro na operação aplicada a um arquivo no disco.

Sintaxe: FERROR( ).

FIELD( )

Propósito: Retornar o nome de um campo do arquivo de dados atual.

Sintaxe: FIELD(<posição>).

Exemplo:

```
USE MALA
? FIELD(2)           // resultado: NOME
FOR I := TO FCOUNT( )
     ? FIELD(I)
NEXT
```

FILE( )

Propósito: Verificar a existência de arquivos gravados no disco.

Sintaxe: FILE(<arquivo>).

Exemplo:

```
USE MALA              // abre o arquivo de dados
IF FILE("NOMEI.NTX")        // se existir o arquivo NOMEI.NTX
     SET INDEX TO NOMEI    // abre o arquivo de índice
ELSE             // se existir
     INDEX ON NOME TO NOMEI          // cria o arquivo de índice
ENDIF
IF FILE("\PRODUÇÃO\CADASTRO.DBF")// verifica no diretório
                                     // \PRODUÇÃO
     .
     .
     .
```

FKLABEL( )

Propósito: Retorna uma cadeia de caracteres representante ao nome da tecla de função especificada.

Sintaxe: FKLABEL(<número da tecla>).

Exemplo:

```
? FKLABEL(5)              // resultado: F5
? FKLABEL(7)              // resultado: F7
```

FKMAX( )

Propósito: Retornar um valor numérico inteiro que representa o número de teclas de funções.

Sintaxe: FKMAX( ).

Exemplo:

```
? FKMAX( )           // resultado: 40
```

FLOCK( )

Propósito: Travar o arquivo de dados quando todos os registros de Redes aberto em modo compartilhado em ambiente Locais.

Sintaxe: FLOCK( ).

Exemplo:

```
USE MALA SHARED            // abre o arquivo compartilhado
     IF FLOCK( )
          ? "ARQUIVO TRAVADO!"
          REPLACE ALL SALÁRIO WITH SALARIO*1.2
     ELSE
          ? "NÃO FOI POSSÍVEL TRAVAR O ARQUIVO"
     ENDIF
```

FOPEN( )

Propósito: Abrir um arquivo binário.

Sintaxe: FOPEN(<arquivo>, <modo>)

FOUND( )

Propósito: Verificar se uma pesquisa no arquivo de dados foi bem sucedida.

Sintaxe: FOUND( ).

Exemplo:

```
USE MALA
LOCATE NOME = "JOÃO"
     IF FOUND( )
          DISPLAY NOME, ENDEREÇO
     ELSE
          ? "REGISTRO NÃO ENCONTRADO"
     ENDIF
```

FREAD( )

Propósito: Realizar a leitura de caracteres de um arquivo, armazenando os mesmos em uma variável.
Sintaxe: FREAD(<handle>, @ <variável>, <bytes>).

Exemplo:

```
# DEFINE BLOCO 128
BUFFER := SPACE(BLOCO)
HANDLE := FOPEN("LEIAME.TXT")
IF FERROR ( ) ! = 0
     ? "NÃO FOI POSSÍVEL ABRIR O ARQUIVO"
ELSE
     IF FREAD (HANDLE, @BUFFER, BLOCO) <> BLOCO
          ? "ERROR NA LEITURA DO ARQUIVO"
     ENDIF
ENDIF
?BUFFER         // mostra os caracteres lidos do um arquivo
```

FREADSTR( )

Propósito: Retorna caracteres lidos de arquivo.
Sintaxe: FREADSTR(<handle>, <bytes>).

Exemplo:

```
# DEFINE "FILEIO.CH"
HANDLE := FOPEN ("LEIAME.TXT", FC_NORMAL)
IF FERROR( ) ! = 0
     ? "ERROR DE ABERTURA"
     CANCEL
ELSE
     STRING
     FCLOSE(HANDLE)
ENDIF
```

FRENAME( )

Propósito: Renomear um arquivo gravado em disco.
Sintaxe: FRENAME(<nome atual>, <novo nome>).

Exemplo:

```
IF FRENAME("LEIAME.TXT", "NAOLEIAM.TXT") <> -1
     ? "ARQUIVO RENOMEADO"
ELSE
     ? "FALHA NA OPERAÇÃO!!!"
ENDIF
```

FSEEK( )

Propósito: Deslocar o ponteiro de arquivo para uma nova posição dentro do mesmo.

Sintaxe: FSEEK(<handle>, <bytes>, <início>).

Exemplo:

```
# INCLUDE "FILEIO.CH"       // diretório \clipper5\include
IF (HANDLE := FOPEN ("LEIAME.TXT")) > 0
     TAMANHO := FSEEK (HANDLE, 0, FS_END)
     // posiciona o ponteiro no início do arquivo
     FSEEK (HANDLE, 0)
ELSE
     ? "NÃO FOI POSSÍVEL ABRIR O ARQUIVO"
ENDIF
```

FWRITE( )

Propósito: Grava uma expressão de caracteres em um arquivo aberto.

Sintaxe: FWRITE(<handle>, <caracteres>, <bytes>).

GETENV( )

Propósito: Carregar o conteúdo de uma variável do sistema operacional DOS.

Sintaxe: GETENV(<variável de ambiente>).

Exemplo:

```
CAMINHO := GETENV ("PATH")   // lê a configuração do PATH do DOS.
SET PATH TO (CAMINHO)      // configura o caminho de pesquisa de
                     // arquivo da aplicação, ajustando-a com o
                     // PATH corrente do DOS.
```

HARDCR( )

Propósito: Substitui todos os Soft Carriage Returns (HR(141), ou seja os retornos automáticos) encontrados em uma expressão caractere por Hard Carriage Returns (CHR(13), ou seja, retornos manuais).

Sintaxe: HARDCR(<expressão caractere>).

Exemplo:

```
USE CLIENTES          // abre o arquivo de dados
SET PRINT ON          // liga a saída do comando de console para
                      // impressora.
? HARDCR(OBS)         // mostra (imprime) o conteúdo do campo memo
                      // formatando a mudança automática de linha
                      // de Memoedit( ).
SET PRINTER OFF       // desliga a impressora.
```

HEADER( )

Propósito: Retornar o número de Bytes do cabeçalho do arquivo de dados em uso.

Sintaxe: HEADER( ).

Exemplo:

```
USE CLIENTES
? HEARDER( )    // mostra o tamanho do cabeçalho do arquivo de
                // dados CLIENTES.DBF.
```

IF( )

Propósito: Processar um teste condicional.

Sintaxe: IF (<condição>, <Ação1>, <Ação2>).

Exemplo:

```
SALDO := 10000.00
R:=IF(SALDO <0, "OK", "SALDO NEGATIVO") // resultado: "OK"
N:= SPACE(30)
@ 10,10 SAY "DIGITE O NOME...:" GET N VALID;
IF(N <> SPACE(30), .T. , .F. )        // analisa a expressão
                                      // digitada por GET
READ
```

INDEXEXT( )

Propósito: Retornar uma string que indica o tipo de arquivo de índice que está sendo processado pelo programa Clipper, ou seja, .NTX ou NDX.

Sintaxe: INDEXEXT( ).

Exemplo:

```
USE MALA              // abre arquivo de dados.
IF .NOT. FILE("INOME" + INDEXEXT( ) )    // verifica se o arquivo
                      // de índice existe, a função substitui a
                      // expressão .NDX OU NTX.
     INDEX ON NOME TO INOME     // caso não exista, é criado.
ENDIF
```

INDEXKEY( )

Propósito: Retornar a expressão de chave de um índice especificado.

Sintaxe: INDEXKEY(<ordem>).

Exemplo:

```
USE CLIENTES INDEX INOME, IENDERECO
? INDEXKEY(2)         // resultado: IENDEREÇO
? INDEXKEY(1)         // resultado: INOME
```

INDEXORD( )

Propósito: Fornecer a ordem de abertura do arquivo de índice corrente.

Sintaxe: INDEXORD( ).

Exemplo:

```
USE CLIENTES
SET INDEX TO INOME, ICEP, ICODIGO
? INDEXORD( )         // resultado: 1
VOLTA := INDEXORD( )
SET ORDER TO 2        // muda o controle do arquivo para ICEP
? INDEXORD( )         // resultado: 2
SET ORDER TO VOLTA
? INDEXORD( )         // resultado: 1
```

INKEY( )

Propósito: Aguarda do buffer do teclado um caractere qualquer.

Sintaxe: INKEY(<tempo>).

Exemplo:

```
@ 22,10 SAY "TECLE ALGO PARA CONTINUAR"
TECLA := INKEY(5)          // espera por um máximo 5 segundos
? 23,01 SAY "VOCÊ PRESSIONOU A TECLA DE CÓDIGO..:" + STR(TECLA)
```

INT( )

Propósito: Retornar o valor inteiro de uma expressão numérica.

Sintaxe: INT(<número>).

Exemplo:

```
VAR1:=VAR2:=2929.93
? INT(VAR1), VAR2          // resultado: 2929    2929.93
```

ISALPHA( )

Propósito: Pesquisar em uma expressão caractere, se o caractere mais à esquerda (primeiro) é uma letra.

Sintaxe: ISALPHA(<expressão caractere>).

Exemplo:

```
VAR1:= "RUA 13 DE MAIO"
VAR2:= "1928"
? ISALPHA(VAR1)          // resultado: .T.
? ISALPHA(VAR2)          // resultado: .F.
```

ISCOLOR( )

Propósito: Pesquisar se o computador que está rodando a aplicação possui a característica de exibir cores.

Sintaxe: ISCOLOR( ).

ISDIGIT( )

Propósito: Pesquisa se o primeiro caractere de uma expressão caractere é um número.

Sintaxe: ISDIGIT(<expressão caractere>).

Exemplo:

```
VAR1:= "RUA 13 DE MAIO"
VAR2:= "1928"
? ISDIGIT(VAR1)      resultado: .F.
? ISDIGIT(VAR2)      resultado: .T.
```

ISLOWER( )

Propósito: Pesquisa se o primeiro caraactere de uma expressão caractere é uma letra maiúscula.

Sintaxe: ISLOWER(<expressão caractere>).

Exemplo:

```
VAR1:= "RUA 13 DE MAIO"
VAR2:= "1928"
? ISLOWER(VAR1)            // resultado: .T.
? ISLOWER(VAR2)            // resultado: .F.
? ISLOWER("EDITORA ERICA")      // resultado: .F.
```

ISPRINTER( )

Propósito: Testar se a impressora conectada na LPT1 está pronta para impressões.

Sintaxe: ISPRINTER( ).

Exemplo:

```
     :
RESPOSTA := "S"
@ 22,10 SAY "CONFIRMA SAÍDA DO RELATÓRIO...:" GET RESPOSTA
READ
     IF .NOT. ISPRINTER( )      // verifica se a impressão não se
                                // encontra pronta.
          @ 23,10 SAY "IMPRESSÃO NÃO PRONTA"
          TONE(300,1)      // emite um som
          INKEY(3)         // aguarda três segundos
          LOOP             // sobe até a linha do DO WHILE
     ENDIF
REPORT FORM RELFOLHA TO PRINT         // saída do relatório.
```

ISUPPER( )

Propósito: Pesquisar se o primeiro caractere de uma expressão caratere é uma letra maiúscula.

Sintaxe: ISUPPER(<expressão caractere>).

Exemplo:

```
VAR1:= "RUA 13 DE MAIO"
VAR2:= "1928"
? ISLOWER(VAR1)       // resultado: .T.
? ISLOWER(VAR2)       // resultado: .F.
```

```
? ISLOWER("EDITORA ERICA")        // resultado: .T.
```

I2BIN( )

Propósito: Realizar a conversão de um número inteiro para inteiro binário de 16 bits.

Sintaxe: I2BIN( ).

LASTKEY( )

Propósito: Retornar o código INKEY( ) da última tecla que foi pressionada.

Sintaxe: LASTKEY( ).

Exemplo:

```
//    SEÇÃO DE @..GETS
READ
      IF LASTKEY( ) = 27    // se a última tecla foi o <ESC>
            RETURN          // termina
      ENDIF
```

LASTREC( )

Propósito: Verificar a quantidade de registros no arquivo de dados corrente.

Sintaxe: LASTREC( ).

Exemplo:

```
USE CLIENTES
? RECCOUNT( ), LASTREC( )        // resultado: 212     212
SET FILTER TO ESTADO = "SP"
? LASTREC( )                     // resultado: 212
COUNT TO TOTALREG
? TOTALREG                       // resultado: 46
```

LEFT( )

Propósito: Extrair um segmento de caracteres retirados do início de uma expressão caractere.

Sintaxe: LEFT(<exp. caractere>, <quantidade>).

Exemplo:

```
VAR := “RUA 13 DE MAIO”
? LEFT(VAR, 3)        // resultado: RUA
```

LEN( )

Propósito: Fornecer o número de elementos de um vetor ou o tamanho de uma expressão caractere.

Sintaxe: LEN(<exp. caractere>|<vetor>).

Exemplo:

LOCAL VAR

```
? LEN(“RUA”)          // resultado: 3
VAR := “RUA 13 DE MAIO”
? LEN(VAR)            // resultado: 14
```

LENNUM( )

Propósito: Fornecer o tamanho de uma expressão numérica.

Sintaxe: LENNUM(<exp. numérica>).

Exemplo:

```
VAR := 299.999
? LENNUM(VAR)              // resultado: 7
```

LOG( )

Propósito: Fornecer o logaritmo natural de uma expressão numérica.

Sintaxe: LOG(<exp.numérica>).

Exemplo:

```
? LOG(10)       // resultado: 2.30
? LOG(2.71)           // resultado: 1.00
```

LOWER( )

Propósito: Converter caractere de maiúsculas para minúsculas.

Sintaxe: LOWER(<exp. caractere>).

Exemplo:

```
NOME := "JOÃO DA SILVA"
? LOWER(NOME)         // resultado: JOÃO DA SILVA.
```

LTRIM( )

Propósito: Remover todos os espaços em branco à esquerda de uma expressão caractere.

Sintaxe: LTRIM(<exp. caractere>).

Exemplo:

```
VALOR := 100
STRING := STR(VALOR)
? STRING              // resultado: 100
? LTRIM(STRING)       // resultado: 100
```

LUPDATE( )

Propósito: Fornecer a data da última atualização do banco de dados corrente.

Sintaxe: LUPDATE( ).

Exemplo:

```
USECLIENTES
? LUPDATE( )          // resultado: DD/MM/AA
```

L2BIN( )

Propósito: Converter um valor inteiro para um inteiro binário de 32 bits.

Sintaxe: L2BIN(<exp.numérica>).

MAX( )

Propósito: Fornecer o maior valor entre duas proporções numéricas ou datas.

Sintaxe: MAX(<exp. numérica 1>, <exp. numérica 2>).
MAX(<data1>, <data 2>).

Exemplo:

```
? MAX (200, 292)      // resultado: 292
```

MAXCOL( )

Propósito: Especificar o número máximo de coluna disponível da tela.

Sintaxe: MAXCOL( ).

Exemplo:

```
@ 0,0 TO MAXROW( ), MAXCOL( ) DOUBLE
```

MAXROW( )

Propósito: Especificar o maior número de linha da tela.

Sintaxe: MAXROW( ).

Exemplo:

```
OBS := SPACE(10)
MEMOEDIT(OBS,0,0, MAXROW( ), MAXCOL( ) )
```

MEMOEDIT( )

Propósito: Permitir a edição na tela de campos memos ou expressões caractere.

Sintaxe: MEMOEDIT (<expressão caractere>, <linha inicial>, <coluna inicial>, <linha final>, <coluna final>, <modo de edição>, <função do usuário>, <tamanho da linha>, <tamanho da tabulação>, <linha buffer>, <coluna buffer>, <linha da janela>).

Exemplo:

```
FALTA EXEMPLO
CLEAR
USE CLIENTES
LOCATE FOR NOME = "JOÃO"          // acesa um registro
@ 2,2 TO 22,79 DOUBLE
TEXTO = SPACE (10)
TEXTO:= MEMOEDIT ( TEXTO,3,3,21,78,.T., "ACENTO")
REPLACE OBS WITH TEXTO            //  salva  texto  no  campo  do
arquivo
//  :
//  :
************************************************************
*     ACENTUAÇÃO DE CAMPOS MEMOS EM CLIPPER 5.0/5.01
*     POR: GORKI STARLIN C. OLIVEIRA
*     DATA: NOV. 92
************************************************************
FUNCTION ACENTO ( MODO, LINHA, COLUNA)     // DEFINIR A FUNÇÃO
# DEFINE INSERT 22                    // cria uma coluna contante
LOCAL RETORNA := LASTKEY( )           // armazena a última tecla
CURSOR := .T.          // modo insert -> .T. (insere)
@ 01,02 SAY "LINHA...:" + STR(LINHA) + "COLUNA...:" + STR(COLUNA)
DO CASE                // faça os casos
     CASE MODO = 3
          CURSOR := .F.          // cursor normal
          SET  CURSOR(  IF(CURSOR,2,1)  )   //  muda  a  forma  do
cursor
          TECLAS( )        // chama a função tecals( )
     CASE MODO = 0   // modo em estado de espera
          IF READINSERT( ) != CURSOR
               SETCURSOR( IF(CURSOR,2,1) )     //  muda  a  forma
do
                                               // cursor
          ENDIF
          TECLAS( )
     OTHERWISE  // <F2> ou <CONTRL-W> grava e sai do memoedit( )
```

```
          IF LASTKEY( ) = - 1        // se teclar <F2>
               RETORNA := 23         // retorna <CTRL>+<W>
          ELSEIF     LASTEKEY( ) = INSERT
                     CURSOR = !READINSERT( )
                     SETCURSOR(IF (CURSOR,2,1) )
                     RETORNA := 22
          ENDIF
DO CASE
     RETURN (retorna)// retorna o controle para DBEDIT( ).

     // ATIVA AS FUNÇÕES DE TRATAMENTO DE ACENTOS
FUNCTION TECLAS            // define a função teclas( ).
     // interliga teclas as funções de tratamento individual de
     // acentos.

SET KEY 39 TO AGUDO( )
SET KEY 96 TO CRAZE( )
SET KEY 94 TO CIRCUNFLEX( )
SET KEY 34 TO TREMA
SET KEY 47 TO CCEDILHA
SET KEY 126 TO TIL
RETURN .T.

FUNCTION AGUDO( )
// Esta função será chamada toda vez que a tecla (‘), ou seja
// CHR(39) for teclada

LOCAL TECLA := INKEY(0)
LOCAL CARACTERE := CHR(TECLA)

     DO CASE
          CASE CARACTER = “A”
               KEYBOAD “A”
          CASE CARACTER = “a”
               KEYBOAD CHR9160)
          CASE CARACTER = “E”
               KEYBOAD CHR(144)
          CASE CARACTER = “e”
               KEYBOAD CHR(130)
          CASE CARACTER $ “iI”
               KEYBOAD CHR(161)
          CASE CARACTER $ “Oo”
               KEYBOAD CHR(162)
          CASE CARACTER $ “Uu”
               KEYBOAD CHR(163)
          OTHER
               SET KEY 39 TO
```

```
                KEYBOAD CARACTER
      ENDCASE
      RETURN .T.

FUNCTION CRAZE
      // Esta função será chamada toda vez que a tecla(^), ou seja
      // CHR(94) for teclada

CARACTERE:= CHR(TECLA)

      DO CASE
            CASE CARACTER = “A”
                  KEYBOAD CHR(143)
            CASE CARACTER = “a”
                  KEYBOAD CHR(131)
            CASE CARACTER = “e”
                  KEYBOAD CHR(136)
            CASE CARACTER $ “Oo”
                  KEYBOAD CHR(162)
            OTHER
                  SET KEY 94 TO
                  KEYBOAD CARACTER
      ENDCASE
      RETURN .T.

      // FUNÇÃO PARA TREMA - ASPAS - CHR(34)

FUNCTION TREMA( )
      // Esta função será chamada toda vez que a tecla(“), ou seja
      // CHR(34) for teclada
      // Para ativar o trema: <“> + <U>

TECLA := INKEY(0)
CARACTERE:= CHR(TECLA)

      DO CASE
            CASE CARACTER = “U”
                  KEYBOARD CHR(154)
            CASE CARACTER = “u”
                  KEYBOARD CHR(129)
            OTHER
                  SET KEY 34 TO
                  KEYBOARD CARACTER
      ENDCASE
      RETURN .T.

      // FUNÇÃO PARA O TRATAMENTO DE CEDILHA - CHR(47) - /
```

```
FUNCTION CCEDILHA( )
     // Esta função será chamada toda vez que a tecla(,), ou seja
     // CHR(47) for teclada
     // Para ativar a cedilha: </> + <C>

TECLA := INKEY(0)
CARACTERE:= CHR(TECLA)

     DO CASE
          CASE CARACTER = "C"
               KEYBOARD CHR(128)
          CASE CARACTER = "c"
               KEYBOARD CHR(135)
          OTHER
               SET KEY 44 TO
               KEYBOAD CARACTER
     ENDCASE
     RETURN .T.

     // FUNÇÃO PARA TRATAMENTO TO ACENTO TIL - CHR(126)

FUNCTION TIL( )
     // Esta função será chamada toda vez que a tecla(~), ou seja
     // CHR(126) for teclada

TECLA := INKEY(0)
CARACTERE:= CHR(TECLA)

     DO CASE
          CASE CARACTER = "A"
               KEYBOAD CHR(142)
          CASE CARACTER = "a"
               KEYBOAD CHR(132)
          CASE CARACTER = "O"
               KEYBOAD CHR(153)
          CASE CARACTER = "o"
               KEYBOAD CHR(148)
          OTHER
               SET KEY 126 TO
               KEYBOAD CARACTER
     ENDCASE
     RETURN .T.
```

MEMOLINE( )

Propósito: Extrair uma linha de texto de um campo memo ou expressão caractere.

Sintaxe: MEMOLINE(<campo>, <tamanho da linha>, <número da linha>, <tamanho da tabulação>, <rolagem>).

Exemplo:

```
//   IMPRESSÃO DE CAMPOS MEMOS
USE CLIENTES
GOTO 3
T_LINHAS := MLCOUNT(OBS,40,3,.T.)     // obtém a quantidade de
                                      // linhas
SET PRINT ON    // liga a saída da console para a impressora
FOR I := 1 TO T_LINHAS
     ? MEMOLINE (OBS,40,I,3,.T.)
NEXT
SET PRINT OFF         // desliga a impressora
```

MEMOREAD( )

Propósito: Carregar o conteúdo de arquivo no formato de texto do disco.

Sintaxe: MEMOREAD(<arquivo>).

Exemplo:

```
// EDITOR DE TEXTOS
CLEAR
VARQUIVO:=SPACE(12)
@ 05,20 SAY "EDITOR DE TEXTOS EM CLIPPER"
@ 10,15 SAY "NOME DO ARQUIVO PARA EDITAR......:" GET VARQUIVO
READ
EDITARQUIVO:= MEMOREAD(VARQUIVO)
@ 00,00 TO 24,79 DOUBLE
EDITARQUIVO:= MEMOEDIT(EDITARQUIVO, 01,01,23,78)
IF .NOT. MEMOWRIT(VARQUIVO,EDITARQUIVO)
     ? "NÃO FOI POSSÍVEL GRAVAR O TEXTO"
ENDIF
```

MEMORY( )

Propósito: Fornecer a quantidade de memória disponível do computador.

Sintaxe: MEMORY(<valor>).

Exemplo:

```
IF MEMORY(2)>= 128    // verificar a disponibilidade de 128 K de
                      // memória livre.
     RUN RELOGIO.EXE
ELSE
     ? "NÃO EXISTE MEMÓRIA DISPONÍVEL"
ENDIF
```

MEMOTRAN( )

Propósito: Substituir os caracteres de controles Carriage return/line feeds em campos Memos ou em uma expressão caractere.

Sintaxe: MEMOTRAN(<campo>, <substituição manual>, <substituição automática>).

Exemplo:

```
REPLACE OBS WITH MEMOTRAN(OBS, " "," ")  // retira todos os
                      // caracteres de controle do texto.
```

MEMOWRIT( )

Propósito: Gravar em um arquivo em disco um campo Memo ou uma expressão caractere.

Sintaxe: MEMOWRIT(<arquivo>, <Memo>).

Exemplo:

```
// EDITOR DE TEXTOS
CLEAR
VARQUIVO:=SPACE(12)
@ 05,20 SAY "EDITOR DE TEXTOS EM CLIPPER"
@ 10,15 SAY "NOME DO ARQUIVO PARA EDITAR....:" GET VARQUIVO
READ
EDITARQUIVO:= MEMOREAD(VARQUIVO)
```

```
@ 00,00 TO 24,79 DOUBLE
EDITARQUIVO:= MEMOEDIT(EDITARQUIVO, 1,1,23,78)
IF .NOT. MEMOWRIT (VARQUIVO, EDITARQUIVO)
     ? "NÃO FOI POSSÍVEL GRAVAR O TEXTO"
ENDIF
```

MIN( )

Propósito: Fornecer o menor valor entre duas datas ou expressões numéricas.

Sintaxe: MIN(<exp. numérica 1>, <exp. numérica 2>).
MIN(<data 1>, <data 2>).

Exemplo:

```
? MIN (300,252)            // resultado: 252
? MIN (DATE( ), DATE( ) + 5)    // resultado: o valor de date( )
```

MLCOUNT( )

Propósito: Fornecer a quantidade de linhas de um campo Memo ou expressão caractere.

Sintaxe: MLCOUNT(<campo>, <tamanho>, <tamanho da tabulação>, <rolagem>).

Exemplo:

```
// IMPRESSÃO DE CAMPOS MEMOS
USE CLIENTES
GOTO 3
T_LINHAS := MLCOUNT(OBS,40,3,.T.)     // obtém a quantidade de
                                      // linhas
SET PRINT ON    // liga a saída da console para a impressora
FOR I := 1 TO T_LINHAS
     ? MEMOLINE(OBAS,40,I,3,.T.)
NEXT
SET PRINT OFF        // desliga a impressora
```

MLPOS( )

Propósito: Fornecer a posição de uma linha de texto dentro de um campo Memo ou uma expressão caractere.

Sintaxe: MLPOS(<campo>, <tamanho da linha>, <número linha>, <tam. tabulação>, <rolagem>).

Exemplo:

```
TEXTO := MEMOREAD("TESTE.TXT")
POSICAO := MLPOS(TEXTO, 40, 5)
? SUBSTR(TEXTO, POSICAO, 15)
```

MOD( )

Propósito: Retorna o resto da divisão do primeiro valor pelo outro.

Sintaxe: MOD(VALOR1, VALOR2).

MONTH( )

Propósito: Realizar a conversão de um dado do tipo Data para um número inteiro referente ao mês da Data.

Sintaxe: MONTH(<data>).

Exemplo:

```
SET DATE TO BRIT
? DATE( )             // resultado: Mostra a data do sistema
? MONTH(DATE( ) )     // resultado: Mostra o número do mês da data
                      // do sistema.
? MONTH(CTOD("11/10/92"))        // resultado: 10
```

NETERR( )

Propósito: Verificar se um comando de manipulação de arquivo de dados falhou em ambiente de Rede Local.

Sintaxe: NETERR( ).

Exemplo:

```
USE MALA SHARED       // abre o arquivo no modo compartilhado
     IF NETERR( )     // se houver erro no comando anterior
          ? "ARQUIVO NÃO DISPONÍVEL"
     ELSE
          SET INDEX TO INOME, ICEP
     ENDIF
    :
    :
```

NETNAME( )

Propósito: Retornar a identificação da estação corrente na Rede Local.

Sintaxe: NETNAME( ).

Exemplo:

```
? NETNAME( )              // resultado: ESTAÇÃO 3
```

NEXTKEY( )

Propósito: Retornar o código (ASCII) do caractere pendente no buffer de teclado.

Sintaxe: NEXTKEY( ).

Exemplo:

```
KEYBOARD "A"               // escreve no buffer o caractere "A"
? NEXTKEY( ), LASTKEY( )        // resultado: 27 27
? INKEY( ), LASKTKEY( )         // resultado: 27 27
? NEXTKEY( )                    // resultado: 0
```

OS( )

Propósito: Fornecer o nome do sistema operacional.

Sintaxe: OS( ).

Exemplo:

```
? OS( )              // resultado: DOS 5.0
```

PAD( )

Propósito: Proporcionar o preenchimento de caracteres em variáveis ou valores dos tipos numérico, data ou caractere.

Sintaxe: PADL(<variável|valor>, <tamanho>, <caractere>)
PADC(<variável|valor>, <tamanho>, <caractere>)
PADR(<variável|valor>, <tamanho>, <caractere>)

Exemplo:

```
DADO1:= "ÉRICA"
DADO2:= "ÉRICA"
DADO1:= "ÉRICA"
? PADC(DADO1,20,"=")        // resultado: =====ÉRICA=======
? PADL(DADO1,20,"=")        // resultado: ============ÉRICA
? PADR(DADO1,20,"=")        // resultado: ÉRICA============
```

PCOL( )

Propósito: Devolver a posição numérica referente à coluna da cabeça de impressão.

Sintaxe: PCOL( ).

Exemplo:

```
SET DEVICE TO PRINT
@ 10,10 SAY DATE( )
@ 10, PCOL( ) +5 SAY TIME( )
SET DEVICE TO SCREEN
```

PCOUNT( )

Propósito: Determinar o número de parâmetros recebidos pela rotina a ser executada.

Sintaxe: PCOUNT( ).

Exemplo:

```
A:= 37
B:= 39
IF COMPARA(A,B)
:
:     <instruções>
:
FUNCTION COMPARA(VALOR1, VALOR2)
IF PCOUNT( ) = 0// se não foi passado nenhum parâmetro
      ? "ERRO, NÃO FORAM PASSADOS OS VALORES PARA COMPARAÇÃO"
      RETURN
ENDIF
IF VALOR1 = VALOR2
      RETURN .T.
ELSE
      RETURN .F.
ENDIF
```

PROCLINE( )

Propósito: Fornecer o número de linhas de um programa fonte.

Sintaxe: PROCLINE(<ativação>).

Exemplo:

```
FUNCTION COMPARA(VALOR1, VALOR2)
IF PCOUNT( ) = 0// se não foi passado nenhum parâmetro
      ? "ERRO, NÃO FORAM PASSADOS OS VALORES PARA COMPARAÇÃO"
      RETURN
ENDIF
? PROCLINE( )         // resultado: 11
IF VALOR1 = VALOR2
      RETURN .T.
ELSE
      RETURN .F.
ENDIF
```

PROCNAME( )

Propósito: Fornecer o nome da rotina que foi ou está sendo executada.

Sintaxe: PROCNAME(<ativação>).

PROW( )

Propósito: Fornecer o número da linha do posicionamento atual da cabeça da impressão.

Sintaxe: PROW( ).

Exemplo:

```
SET DEVICE TO PRINT
@ 10, 10 SAY DATE( )
@ PCOL( ) +1,10 SAY TIME( )     // imprime a hora na linha 11
SET DEVICE TO SCREEN
```

QOUT( )

Propósito: Exibe uma lista de expressões no console.

Sintaxe: QOUT(<lista de expressões>).

Exemplo:

```
QOUT (DATE( ), TIME( ) )
QOUT ("EDITORA ERICA")
```

QQOUT( )

Propósito: Exibe uma lista de expressões no console na mesma linha.

Sintaxe: QQOUT(<lista de expressões>).

Exemplo:

```
QQOUT (DATE( ), TIME( ) )
QQOUT ("EDITORA ERICA")
```

RAT( )

Propósito: Retorna o endereço da última ocorrência de uma expressão dentro de uma string.

Sintaxe: RAT(<expressão caractere>, <string>).

Exemplo:

```
? RAT("RJ", "SP_RJ_RS,PA")
```

READEXIT( )

Propósito: Ligar ou desligar as teclas de seta para cima e seta para baixo como saídas de um READ.

Sintaxe: READEXIT(<.T.>|<.F.>).

Exemplo:

```
CLEAR
READEXIT(.T.)              // liga a saída através das teclas
ENDERECO:= NOME:= SPACE(30)
@ 10,10 SAY "NOME...........:" GET NOME
@ 12,10 SAY "ENDEREÇO.......:" GET ENDEREÇO
READ
```

READINSERT( )

Propósito: Ligar ou desligar o modo de inserção.

Sintaxe: READINSERT(<.T.>|<.F.>).

Exemplo:

```
CLEAR
ENDERECO:= NOME:= SPACE(30)
@ 23,10 SAY "TECLE <F2> PARA INSERÇÃO"
@ 10,10 SAY "NOME........:" GET NOME
@ 12,10 SAY "ENDEREÇO....:" GET ENDEREÇO
SET KEY -1 TO INSERE( )
```

```
READ
FUNCTION INSERE( )
IF READINSERT( )      // insert ligado
     READINSERT(.F.)
     @ 00,65 SAY "   "
ELSE
     READINSERT(.T.)
     @ 00, 65 SAY "INSERE"
ENDIF
RETURN
```

READKEY( )

Propósito: Fornecer o número da última tecla que encerrou um READ.

Sintaxe: READKEY( ).

READMODAL( )

Propósito: Executar uma lista de GET's.

Sintaxe: READMODAL(<vetor de GET's>).

READVAR( )

Propósito: Devolver o nome da variável GET ou MENU TO atual.

Sintaxe: READVAR( ).

Exemplo:

```
CLEAR
SET KEY -2 TO TESTE( )   // tecla F2 para rodar a função teste( )
NOME:= ENDERECO:= SPACE(30)
@ 10,10 SAY "NOME.....................:" GET NOME
@ 12,10 SAY "ENDEREÇO.................:" GET ENDEREÇO
READ
FUNCTION TESTE( )
IF READVAR + "NOME"
     ? "VOCÊ ESTA PROCESSANDO A VARIÁVEL DO NOME"
ELSEIF READVAR = "ENDEREÇO"
```

```
     ? "VOCÊ ESTA PROCESSANDO A VARIÁVEL DO ENDEREÇO"
ENDIF
RETURN
```

RECCOUNT( )

Propósito: Retornar o número de registro do arquivo de dados atual.

Sintaxe: RECCOUNT( ).

Exemplo:

```
USE MALA
? RECCOUNT( )              // resultado: 115
COUNT TO TOTALREG
? TOTAL REG                // resultado: 115
SET DELETE ON              // omite os registro deletados
? RECCOUNT( )              // resultado: 115
COUNT TO TOTALREG
? TOTAL REG                // resultado: 94
```

RECNO( )

Propósito: Retorna o número do registro corrente.

Sintaxe: RECNO( ).

Exemplo:

```
USE MALA
GO 1
? RECNO( )           // resultado: 1
SKIP + 3
? RECNO( )           // resultado: 4
```

RECSIZE( )

Propósito: Devolver o tamanho em Bytes do registro do arquivo.

Sintaxe: RECSIZE( ).

Exemplo:

```
USE MALA
? “TAMANHO TOTAL DO ARQUIVO”
?? RECSIZE( ) * LASTREC( ) + HEADER( )
```

REPLICATE( )

Propósito: Retornar uma cadeia de caracteres contendo a repetição de um ou mais caracteres.

Sintaxe: REPLICATE(<caractere(s)>, <quantidade>).

Exemplo:

```
? REPLICATE(“=“, 5)              // resultado: =======
? REPLICATE(“CASA”, 3)      // resultado: CASACASACASA
```

RESTSCREEN( )

Propósito: Restaurar uma área de tela anteriormente gravada em uma região específica do vídeo.

Sintaxe: RESTSCREEN (<L_inicial>, <C_inicial>, <L_Final>, <C_Final>, <tela>).

Exemplo:

```
CLEAR
@ 10,10 TO 20,70 DOUBLE          // desenha uma moldura
A:= SAVESCREEN(10,10,20,70)      // salva a tela
CLEAR                            // limpa a tela
RESTCREEN(10,10,20,70,A)
```

RIGHT( )

Propósito: Devolver uma substring de uma string a partir de uma determinada posição à direita.

Sintaxe: RIGHT(<string>, <posição>).

Exemplo:

```
? RIGHT (“EDITORA ERICA”, 5)          // resultado: ERICA
```

RLOCK( )

Propósito: Travar o registro corrente do banco de dados em ambiente de Rede Local.

Sintaxe: RLOCK( ).

Exemplo:

```
USE MALA SHARED
SEEK “JOÃO”
     IF RLOCK( )
          DELETE
          ? “REGISTRO DELETADO!!!”
     ELSE
          ? “REGISTRO NÃO PODE SER TRAVADO (ALTERADO) NESTE;
          MOMENTO”
     ENDIF
```

ROUND( )

Propósito: Arredondar expressões numéricas.

Sintaxe: ROUND(<valor numérico>, <casas decimais>).

Exemplo:

```
? ROUND(103.338556,2)      // resultado:   103.34
? ROUND(103.33855, 0)      // resultado:   103.00
```

ROW( )

Propósito: Devolver o número da linha em vídeo na qual o cursor se encontra posicionado.

Sintaxe: ROW( ).

Exemplo:

```
CLEAR
@ 10,10 SAY “TESTE NA LINHA 10”
@ ROW( ) + 3, 10 SAY “TESTE NA LINHA 13”
@ ROW( ) - 1, 10 SAY “TESTE NA LINHA 12”
@ 20,10 SAY “TESTE NA LINHA 20”
@ ROW( ) + 1, 10 SAY “TESTE NA LINHA 21”
```

RTRIM( )

Propósito: Remover espaços em branco existentes no final de uma expressão caractere.

Sintaxe: RTRIM(<exp. caractere>).

Exemplo:

```
CEP :=”11020202”
ESTADO := “SP” + SPACE(10)        // estado  = “SP”
? ESTADO + CEP              // resultado : SP    11020202
? RTRIM(ESTADO)+CEP         // resultado: SP 11020202
```

SAVESCREEN( )

Propósito: Salvar uma área especifica da tela.

Sintaxe: RESTCREEN (<L_inicial>, <C_inicial>, <L_Final>, <C_Final>, <tela>).

Exemplo:

```
CLEAR
@ 10,10 TO 20,70 DOUBLE          // desenha uma moldura
A := SAVESCREEN(10,10,20,70)          // salva a tela
? “TECLE ALGO”
INKEY(0)
CLEAR                        // limpa a tela
RESTSCREEN(10,10,20,70,A)  // restaura a tela anteriormente salva
```

SCROLL( )

Propósito: Rolar uma área da tela para baixo ou para cima.

Sintaxe: SCROLL (<L_inicial>, <C_inicial>, <L_Final>, <C_Final>, <linhas a rolar>).

Exemplo:

```
FUNCTION MOSTRAROLA (L_I, C_I, L_F, C_F,)
SCROLL (L_I, C_I, L_F, C_I, 1)  // rola um linhas para cima
@ L_F, C_I SAY DADO
RETURN
```

SECONDS( )

Propósito: Devolver a quantidade em segundos da hora atual do sistema.

Sintaxe: SECONDS( ).

Exemplo:

```
? TIME( )                  // resultado: 10:00
? SECONDS( )               // resultado: 36000
```

SELECT( )

Propósito: Devolver o número da área de trabalho especificada.

Sintaxe: SELECT(<nome da área>).

Exemplo:

```
USE MALA NEW
? SELECT( )                // resultado: 1
USE CLIENTES NEW
? SELECT( )                // resultado: 2
? SELECT("MALA")       // resultado: 1
```

SETCANCEL( )

Propósito: Ligar/desligar a tecla ALT-C.

Sintaxe: SETCANCEL(.T.|.F.).

Exemplo:

```
/*    PROGRAMA: PRINCIPAL.PRG
      AUTOR: GORKI STARLIN
*/
SET DELETE ON
SET CONFIRM ON
SETCANCEL(.F.)        // desliga a saída via ALT-C.
```

SETCOLOR( )

Propósito: Setar novas cores do vídeo ou verificar a configuração atual das mesmas.

Sintaxe: SETCOLOR(<cor nova>).

Exemplo:

```
ATUAL  := SETCOLOR( )       // armazena as cores atuais em uma
                            // variável.
PADRÃO := "BR+/N"
DESTAQUE := "R+/N"
SET COLOR(PADRÃO, DESTAQUE)      // configura um novo padrão de
                                 // cores.
```

SETCURSOR( )

Propósito: Configurar a forma do cursor.

Sintaxe: SETCURSOR(<valor>).

Exemplo:

```
CLEAR
NOME := ENDEREÇO := SPACE(30)         // inicializa as variáveis
SETCURSOR(1)
@ 10,10 SAY "NOME.................:" GET NOME
SETCURSOR(3)
@ 12,10 SAY "ENDEREÇO.............:" GET ENDEREÇO
READ
```

SETKEY( )

Propósito: Associar um bloco de código (Code Block) a uma determinada tecla.

Sintaxe: SETKEY(<código da tecla>, <code block>).

SETPOS( )

Propósito: Determinar uma nova posição para o cursor na tela.

Sintaxe: SETPOS(<linha, coluna>).

Exemplo:

```
? SETPOS(10,20)
? "TESTE"
? SETPOS(20,10)
? "TESTE"
```

SETPRC( )

Propósito: Configurar os valores de PROW( ) e PCOL( ).

Sintaxe: SETPRC(<linha>, <coluna>).

Exemplo:

```
SET DEVICE TO PRINTER
SETPRC(10,15)
@ PROW( ), PCOL( ) SAY "TESTE"   // na linha 10 coluna 15 do
                                 // "papel".
```

SOUNDEX( )

Propósito: Converter uma expressão caractere para uma expressão fonética.

Sintaxe: SOUNDEX(<expressão caractere>).

Exemplo:

```
USE MALA
INDEX ON SOUNDEX(NOME) TO INOME.NTX
SEEK SOUNDEX (“JOÃO”)       // pesquisa a fonética do nome JOÃO
IF FOUND( )
     DISPLAY NOME, ENDEREÇO, CIDADE
ELSE
     ? “NÃO ENCONTRADO”
ENDIF
```

SPACE( )

Propósito: Gerar espaços em branco.
Sintaxe: SPACE(<tamanho>).

Exemplo:

```
NOME := SAPCE( )// armazena 30 espaço dentro da variável
? “TESTE! + SPACE(20) + “TESTE2”
```

SQRT( )

Propósito: Devolver a raiz quadrada de uma expressão numérica.
Sintaxe: SQRT(<expressão numérica>).

Exemplo:

```
? SQRT(2)            // resultado:   1.41
? SQRT(4)            // resultado:   2.00
```

STR( )

Propósito: Converter uma expressão numérica em uma expressão caractere.
Sintaxe: STR(<valor numérico>, <comprimento>, <casas decimais>).

Exemplo:

```
SALÁRIO := 3020.29
? SRT(SALARIO,4)      // resultado:   3030
? SRT(SALARIO,7,3)          // resultado:   3020.290
```

STRTRAN( )

Propósito: Localiza e trocar caracteres de uma string ou campo memo.

Sintaxe: STRTRAN(<string>, <caracteres>, <substituição>, [<posição início>], [<quantidade>]).

Exemplo:

```
TEXTO:=  "CLIPPER 5.X E CLIPPER 87"
? STRTRAN(TEXTO,"CLIPPER", "VERSÃO")    // resultado: VERSÃO 5.0 E
                                        // VERSÃO 87
```

STUFF( )

Propósito: Anexar e retirar caracteres de uma string.

Sintaxe: STUFF(<string>, <início>, <retirar>, <anexar>).

Exemplo:

```
// somente inserir
TESTE := "JCARLOS"
? STUFF(TESTE1, 2, 0, "OAO")          // resultado: JOÃO CARLOS
// extrair e depois inserir
? STUFF(TESTE1, 2, 6, "OAO")          // resultado: JOÃO
```

SUBSTR( )

Propósito: Extrair uma seqüência de caracteres de uma string.

Sintaxe: SUBSTR(<string>, <posição inicial>, <tamanho>).

Exemplo:

```
TESTE1 := "JOÃO CARLOS"
? SUBSTR(TESTE1, 1, 4)          // resultado: JOÃO
? SUBSTR(TESTE1, 5, 7)          // resultado: CARLOS
```

TIME( )

Propósito: Devolver a hora do sistema.

Sintaxe: TIME( ).

Exemplo:

```
CLEAR
@ 10,10 SAY "HORA ATUAL......:" + TIME( )
```

TONE( )

Propósito: Executar uma freqüência sonora com uma duração especificada.

Sintaxe: TONE(<freqüência>, <duração>).

Exemplo:

```
BEEP( )
FUNCITION BEEP
TONE(300,18)
TONE(200,26)
```

TRANSFORM( )

Propósito: Transformar qualquer valor para uma cadeia de caracteres com um formato definido.

Sintaxe: TRANSFORM(<valor>, <formato>).

Exemplo:

```
? TRANSFORM("EDITORA ERICA", "@!")      // resultado: EDITORA ERICA
```

TYPE( )

Propósito: Identificar qual o tipo de uma informação.
Sintaxe: TYPE(<dado>).

Exemplo:

```
VALOR := 1919
NOME  := "JOÃO"
BLOCO := {|| L++}
? TYPE("VALOR")            // resultado: N
? TYPE("NOME")             // resultado: C
? TYPE("BLOCO")            // resultado: B
```

UPDATED( )

Propósito: Verificar se os Get's foram alterados durante a edição em tela.
Sintaxe: UPDATED( ).

Exemplo:

```
USE MALA
DO WHILE .T.
CLEAR
@ 07,15 SAY "CADASTRAMENTO"
VNOME := SPACE(30)
VENDERECO := SPACE(40)
@ 10,10 SAY "NOME..........:" GET VNOME PICT "@!"
@ 12,10 SAY "ENDEREÇO.......:" GET VENDERECO PICT "@!"
READ
     IF .NOT. UPDATED( )        // se não foi alterado
          LOOP                  // sobe ao DO WHILE
     ENDIF
APPEND BLANK
REPLACE NOME WITH VNOME
REPLACE ENDEREÇO WITH VENDERECO
ENDDO
```

UPPER( )

Propósito: Transforma as letras de uma string para maiúsculas.
Sintaxe: UPPER(<string>).

Exemplo:

```
NOME := "EDITORA ERICA"
? UPPER(NOME)          // resultado:   EDITORA ERICA
@ 20,20 SAY "NOME........:" + UPPER(NOME)
```

USED( )

Propósito: Verificar se existe arquivo de dados (.DBF) na área de trabalho selecionada.

Sintaxe: USED( ).

Exemplo:

```
USE MALA NEW           // abre o arquivo de dados MALA
USE CLIENTES NEW// abre o arquivo de dados CLIENTES
? USED( )              // resultado: .T. (CLIENTES.DBF)
SELECT MALA            // seleciona a área de trabalho do MALA
? USED( )              // resultado: .T. (MALA.DBF)
USE                    // fecham o arquivo de dados mala
? USED( )              // resultado: .F. (NÃO EXISTE .DBF NESTA
                                     ÁREA DE TRABALHO)
```

VAL( )

Propósito: Converter uma expressão caractere em um valor numérico.

Sintaxe: VAL(<string>).

Exemplo:

```
SALÁRIO := "2929.20"
? VAL (SALÁRIO) * 2
TESTE := "COMPUTADOR"
? VAL(TESTE)           // resultado: 0
```

VALTYPE( )

Propósito: Especificar o tipo do dado retornado por uma expressão.

Sintaxe: VALTYPE(<dado>).

Exemplo:

```
? VALTYPE(STR( ) )          // retorna: "C"
? VALTYPE(SQRT( ))          // retorna: "N"
```

```
? VALTYPE(2929)            // retorna: "N"
? VALTYPE("JOÃO")          // resultado: "C"
```

VERSION( )

Propósito: Identificar a versão da linguagem Clipper.
Sintaxe: VERSION( ).

Exemplo:

```
? VERSION( )          // resultado:     Clipper 5.2
```

WORD( )

Propósito: Transformar um parâmetro que será passado para um comando CALL de DOUBLE para INT.
Sintaxe: WORD(<valor numérico>).

Exemplo:

```
CALL progC WITH WORD (10200),   "um texto qualquer"
```

YEAR( )

Propósito: Devolver o ano de uma data.
Sintaxe: YEAR(<data>).

Exemplo:

```
SET DATE BRIT
? YEAR (DATE( ) )          // resultado:   93
DATA := CTOD ("20/12/93")
? YEAR (DATA)         // resultado:   93
```