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 video
? 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
      FUNCÃO:    ACESSAR TODOS OS PROGRAMAS
*/
SAVE SCREN TO TECLADOS
CLEAR
SET DATE BRIT
SET CONFIRM ON
SET DETELE 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 variaveis
:
:
          // formata a digitação para maiusculas
@ 12,10 say “Nome do funcionario.....:” get Vnome pict “@!”
          // edita os numeros no formato europeu
@ 14,10 say “Salario Mensal....:” get vsalario pict “@E
999,999,999.99”
vdata : = date( )    // cria a variavel 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 area da tela de apenas 20
posições,
          // rolando no sentido horizontal o que não couber no
20
          // espaços determinados por PICTURE “@s20”.
@ 18,10 say “Endereco..:” get vendereco picture “@s20”
READ       // executa o get pentende.
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
          // variavel.
```

```
@ 12,15 say “Codigo....:” 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 saida 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 INFORMATICA LTDA.”
@ 03,60 SAY DATE( )
@ 03,70 SAY TIME( )
          // detalha o menu de barras
@ L,C     PROMPT ”INCLUSÃO”          MESSAGE “INCLUSAO DE DADOS”
@ L+1,C   PROMPT “ALTERAÇAO”   MESSAGE “ALTERAÇAO DE DADOS”
@ L+2,C   PROMPT “CONSULTA”    MESSAGE “CONSULTA DE DADOS”
@ L+3,C   PROMPT “EXCLUSAO”    MESSAGE “EXCLUSAO DE DADOS”
@ L+4,C   PROMPT “RELATORIOS”  MESSAGE “RELATORIOS DO SISTEMA”
@ L+5,C   PROMPT “UTILITARIOS” MESSAGE “UTILITARIOS 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
            FUNÇAO: 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 “SIRIOS INFORMATICA”
      @ 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 FUNCIONARIOS **”
@ 08,10 SAY “CODIGO...........:” GET CODVAR PICTURE “9999”
READ
IF CODVAR = 0   // verifica se o usuário nao digitou o codigo
      OP: = “S” // cria a variavel   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 indice o conteudo da variavel
                        // CODVAR
IF EOF( )         // se não existe
      APPEND BLANK      // tenta criar um registro em branco
                        // entra com o restante dos dados do funcionario
```

```
@ 10,10 SAY “NOME FUNCIONARIO..:” GET NOMEVAR PICTURE “@!”
@ 12,10 SAY “SETOR TRABALHO....:” GET SETORVAR PICT “@9”
@ 14,10 SAY “CARGO FUNCIONAL...:” GET CARGOVAR PICT “@!”
@ 16,10 SAY “SALARIO...........:” GET SALARIOVAR PICT “9999999.99”
@ 18,10 SAY “FUNCIONARIO ATIVO.:” GET ATIVOVAR
@ 20,10 SAY “DATA ADMISSAO.....:” 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 SALARIO WITH SALARIOVAR
      @ 21,20 SAY “** CADASTRO **”
      WAIT “ “          // aguarda QQ tecla
      COMMIT            // atualiza fisicamente o registro
  ELSE            // se não
      @ 21,20 SAY “** REGISTRO JA 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 SALARIO, COMISSAO 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
funcionarios
                                // que trabalhem no setor A.
```

BEGIN SEQUENCE

Propósito: Define uma sequência 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 exixtirá nenhuma pendencia 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
                          //(ÁLIAS) 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 indice
    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 VARIAVEIS
CODVAR := SETORVAR :=  SALARIOVAR := 0
NOMEVAR     := SPACE(35)
CARGOVAR    := SPACE(15)
ATIVOVAR    := (.T.)
DATAVAR     := CTOD ("  /  /  ")
            // entrada de dados
@ 06,10 SAY "** CADASTRAMENTO DE FUNCIONARIOS **"
@ 08,10 SAY "CODIGO...........:" GET CODVAR PICTURE "9999"
READ
IF CODVAR = 0     // verifica se o usuário nao digitou o codigo
      OP := "S"   // cria variavel  OP
                  // pergunta se o usuario 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 indice o conteudo da variavel
                        // variavel CODVAR
IF EOF( )         // se NAO EXISTE
      // entra com o restante dos dados do FUNCIONARIO
@ 10,10 SAY "NOME FUNCIONARIO..:" GET NOMEVAR PICTURE "@!"
@ 12,10 SAY "SETOR TRABALHO....:" GET SETORVAR PICT "@9"
@ 14,10 SAY "CARGO FUNCIONAL...:" GET CARGOVAR PICT "@!"
@ 16,10 SAY "SALARIO...........:" GET SALARIOVAR PICT "9999999.99"
@ 18,10 SAY "FUNCIONARIO ATIVO.:" GET ATIVOVAR
@ 20,10 SAY "DATA ADMISSAO.....:" 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 SALARIO 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 JA 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, SALARIO, 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 aquivo 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, SALARIO, COD           // mostra os registros
INKEY(0)
SET DELETE ON           // filtra os registros marcados
DISPLAY ALL NOME, SALARIO, COD           // mostra os registros
INKEY(0)
RECALL ALL              // recupera todos os registros
DISPLAY ALL NOME, SALARIO, COD           // mostra os registros
INKEY(0)
DELETE FOR SETOR = “A”       // marca os funcionários do setor A
DISPLAY ALL NOME, SALARIO, 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 extenção .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, SALARIO ALL           // mostra todos os registros
DIPLAY COD, NOME, SALARIO          //   mostra   somente   o   registro
corrente
DISPLAY COD, NOME, SALARIO 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 parâmentros>]

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
   │           ? "REALATORIO DE FUNCIONARIOS"
   │           ?
   │           ? "Pagina:"+str(pg)
   │           replicate ("=", 78)      // traça uma linha
   │           1 := 7
   │        ENDIF
   │        ? COD, NOME, SALARIO         // 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/procediemento>
[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 SAIDA( )              / / 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 CODIGO...:" 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, ENDERECO, 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)
      :
      :
      :
FUNTION 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, SALARIO
GO TOP     // vá para o inicio do arquivo
DISPLAY NOME, COD, SALARIO
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
          ?   “APROVAADO”
     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 simbolos> [IN <alias>]]
          [LOCAL <simbolos> [: = valor]]
          [MEMVAR <lista de simbolos>
               <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( )       // rotin de inicialização
nSEGUNDOS := SECONDS( )
RETURN
EXIT PROCEDURE SAIDA( )         // rotina de saida 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 exeistir
     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
CELAR
INPUT “DIGITE QUALQUER COISA..:” TO VAR
? “VOCÊ DIGITOU...:”
?? VAR
```

JOIN

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

Sintaxe: `JOIN WITH <2º 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 COMISSAO 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 tres 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 <expessão>
[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 serm 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  SERA 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 := "JOAO"
VENDE:= "RUA DAS CAMELIAS 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 CREEN [TO **<**tela**>**]

SEEK

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

Sintaxe: SEEK **<**chave**>**

Exemplo:

```
USE MALA INDEX INOME
SEEK “JOAO”               // Equivalente A: DBSEEK (“JOAO”)
     IF FOUND( )          // se existir
          DISPLAY NOME, ENDERECO, CIDADE
     ELSE
          ? “NAO 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, SALARIO, SETOR, COD
SELECT MALA     // selecina o arquivo área MALA
LIST COD, CLIENTE, CIDADE

LIST MALA → CLEINTE, FOLHA → SALARIO// 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
                          // digitos 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
? “VOCE 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 destivar a edição de caracteres que serão utilizados como delimitadores de GET's.

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

SET DELIMITER TO

Propósito: Define dlimitadores 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 novomente
@ 12,10 SAY "DIGITE O ENDERECO..:" 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 "SERA 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.

Sinatxe: 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, ENDERECO, CIDADE
SET ORDER TO 2        // ICEP, NOME
LIST NOME, ENDERECO, 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, ENDERECO, 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, ENDERECO, CIDADE, CEP     // lista em ordem de NOMES
SET ORDER TO 2            //  muda  o  arquivo  de  índice  de
controle
LIST NOME, ENDERECO, 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.

Sinatxe: SET PROCEDURE TO < nome do arquivo>

SET RELATION

Propósito: Estabelecer relacionamentos entre áreas de trabalho.

Sinatxe: 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 exebição das mansagens 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 deligar 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 mantado 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, ENDERECO, 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 SALARIO TO TOTALSAL FOR SETOR = 1     // tolaliza o salario
                                  // 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.ext> <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 SALARIO WITH VSAL*INDICE ALL
             UNLOCK            // libera o travamento pendente
          ELSE
               ? "NAO É POSSIVEL 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.
```