ÍNDICE

Aula 01 - O que é o Clipper?

Topo

O CLIPPER é atualmente uma das linguagens de programação mais utilizada no
mercado nacional basicamente por dois motivos:

1.É a mais fácil de ser aprendida.
2.É a que tem melhores recursos para desenvolver menus e indexar arquivos.

Visualize o CLIPPER como uma língua estrangeira. Suponha que você estivesse a
oportunidade de ir para um país estrangeiro e permanecesse lá, por conta própria. Um
dos primeiros requisitos para a sobrevivência seria aprender o idioma local. Não
haveria método melhor para se fazer isso do que ouvir os nativos e praticar a língua
com eles.

Ler livros sem orientação e exemplos práticos não é diferente de tentar aprender uma
língua estrangeira num ambiente de sala de aula, onde sua sobrevivência não depende
da compreensão. Para muitos indivíduos, compreender o CLIPPER é vital para
sobreviver no mundo dos microcomputadores. Este mundo se baseia na velocidade,
na portabilidade e no dinheiro.

Bem, agora vamos direto para o mundo do CLIPPER. Prepare-se para uma linguagem
finalmente movida a crença de que tudo é possível para um microcomputador.

Começando o Trabalho

Topo
Para começar a criar o programa será necessário usar um editor de textos, nós
usaremos o EDIT do DOS por ser muito fácil de ser usado. Também deve ser dado
um nome para o programa, este nome pode ter até 8 caracteres mais uma extensão
.PRG.

Ex.:

EDIT exerc1.prg

Primeiros Comandos

Topo

CLEAR SCREEN

Apaga toda a tela e reposiciona o cursor no início da tela.

@ L1,C1 TO L2,C2 <DOUBLE>

Cria uma janela na tela baseado nas coordenadas especificadas. L1 e L2 variam de 0 a
24 e C1 e C2 variam de 0 a 78. Se for acrescentada a opção DOUBLE a janela sairá
com barras duplas.

@ L1,C1,L2,C2 BOX <STRING>

O comando BOX é usado principalmente para desenhar quadros de maneira rápida e
eficiente, principalmente na implementação de molduras para desenhos de tela. O
parâmetro string do comando BOX precisa ser um string (cadeia de letras) de nove
caracteres ou um string nulo. Se o string nulo for passado ao comando BOX, a área
especificada pelas quatro coordenadas será limpa. Caso contrário, o string que é
passado representa os caracteres destinados a:

(1) canto superior esquerdo, (2) linha do alto, (3) canto superior direito, (4) linha de
baixo do lado direito, (5) canto inferior direito, (6) linha de baixo, (7) canto inferior
esquerdo, (8) linha de cima do lado esquerdo e (9) caractere que irá preencher o
espaço emoldurado pelo quadro.

INKEY(NÚMERO)

Esta função espero "número segundos" para continuar a execução do programa. Se for
colocado o número "0" a função espera até que uma tecla qualquer seja pressionada.

Ex.:

```
CLEAR SCREEN
BORDA = CHR(201) + CHR(205) + CHR(187) + CHR(186) + ;
CHR(188) + CHR(205) + CHR(200) + CHR(186) + ;
CHR(176)
@ 1,1 TO 20,78 DOUBLE
@ 5,5,15,73 BOX BORDA
INKEY(0)
```

Depois de feito esse programa grave-o e saia do Edit.

Executando o programa

Topo

1º - Verificando se não há erros no programa:

```
CLIPPER NOME <ENTER>
```

2º - Vamos agora montar o programa executável.

```
RTLINK FI NOME <ENTER>
```

3º - Vamos agora executar o programa. Para isso basta digitar no prompt do DOS o
nome do programa.

```
NOME <ENTER>
```

Exercícios da Aula 01

Topo

Borda simples:

```
SIMPLES= CHR(218) + CHR(196) + CHR(191) + CHR(179) +;
CHR(217) + CHR(196) + CHR(217) + CHR(179) +;
CHR(#)
```

Borda dupla:

```
DUPLA = CHR(201) + CHR(205) + CHR(187) + CHR(186) + ;
CHR(188) + CHR(205) + CHR(200) + CHR(186) + ;
CHR(#)
```

1º - Escreva um programa que desenhe na tela um quadrado preenchido com a letra
"S". Atenção, o quadrado deverá ter barras simples.

2º - Escreva um programa que desenhe na parte de cima da tela um quadrado
preenchido com a letra "S" e na parte de baixo, um quadrado preenchido com a letra
"B". Atenção, o quadrado de cima deverá ter barras duplas e o de baixo barras simples.

Aula 02 - Comando para a entrada de dados via teclado

Topo
Até agora nós somente tratamos de como criar janelas na tela, agora nós iremos
tratar de criar programas que peçam informações ao usuário. Para isso vai ser preciso
esclarecer o que é uma variável. Uma variável é um determinado valor que nós
armazenamos na memória, nós damos um nome para essa variável, existem vários
tipos de variáveis:

Numérica: possui somente números.
String: contém palavras, tipo nome, endereço, telefone.
Lógica: pode conter apenas dois valores: Verdadeiro ou falso.
Data: possui datas.
Vamos agora aprender a criar essas variáveis:
num=0
Isso cria uma variável numérica com o nome num
nome="XLK Informática"
ou
nome=SPACE(40)

No 1º exemplo é criada uma variável nome com o conteúdo "XLK Informática" e no
segundo exemplo será criada uma variável com quarenta espaços em branco dentro
dela, é muito utilizado essa instrução SPACE.

Verdade=.T.
ou
Falso=.F.

No primeiro exemplo, cria uma variável lógica com o conteúdo .T. (verdadeiro) e no
segundo é criada uma variável com o conteúdo .F. (falso).
Data= CTOD(" / / ")
Cria uma variável com a data a ser preenchida no seu interior.
Como vimos, é dessa forma que se cria variáveis, mas e se o usuário quiser entrar
com seus dados? É isso o que nós faremos agora.
Antes de um usuário entrar com seus dados, ele terá que saber o que que ele está
fazendo, certo? Então vamos escrever uma mensagem na tela para isso usando o
comando SAY.

```
@ L1,C1 SAY "MENSAGEM OU VALOR"
```

Mostra a mensagem entre aspas na posição do vídeo L1,C1
Mostrada a mensagem agora o usuário vai digitar alguma informação, para isso usa-se
o GET

```
@ L1,C1 GET <VARIÁVEL>
```

"Pega" a informação do usuário em uma determinada linha e coluna L1,C1
O Comando GET pode ser usado em conjunção com o comando SAY de forma que
um continue o outro, muito útil devido as linhas e colunas, assim a L1, C1 do GET vai
estar sempre depois do fim da mensagem do SAY:

```
@ L1,C1 SAY "MENSAGEM OU VALOR" GET <VARIÁVEL>
```

O Clipper é muito flexível em relação aos GET's, pode-se criar uma série de GET's e
pode-se passar de um para o outro da forma que se quiser, mas no fim dos GET's é
necessário a instrução READ:

```
READ
```

Outra função usada na criação de telas é:

```
@ L1,C1 CLEAR TO L2,C2
```

Ela limpa a tela de L1,C1 até L2,C2.

Ex.:

```
CLEAR SCREEN
BORDA = CHR(201) + CHR(205) + CHR(187) + CHR(186) + ;
CHR(188) + CHR(205) + CHR(200) + CHR(186) + ;
CHR(176)
@ 1,1,20,78 BOX BORDA
@ 03,03 TO 18,76
@ 04,04 CLEAR TO 17,75
VPRODUTO = SPACE(30)
VCUSTO = 0
VVENDA = 0
VLUCRO = 0
@ 06,06 SAY "NOME DO PRODUTO: "
@ 08,06 SAY "PREÇO DO PRODUTO: "
@ 10,06 SAY "PREÇO DE VENDA: "
@ 12,06 SAY "LUCRO DA VENDA: "
@ 06,23 GET VPRODUTO PICT "@!"
@ 08,23 GET VCUSTO PICT "99,999,999.99"
@ 10,23 GET VVENDA PICT "99,999,999.99"
READ
VLUCRO = VVENDA-VCUSTO
@ 12,23 SAY VLUCRO PICT "99,999,999.99"
```

Exercícios da aula 02

Topo
1º - Baseado no último exemplo, escreva um programa que pergunte o nome do
alunos e as 3 notas. Calcule a média e mostre o resultado.

2º - Escreva um programa que entre com 2 números. Calcule a soma dos dois
números, a subtração, a divisão e a multiplicação.

Aula 03 - Comandos de Condição e Repetição

Topo

IF <CONDIÇÃO> <COMANDOS> ELSE <COMANDOS> ENDIF

O comando IF permite um desvio, dependendo da condição em um ambiente de
programação estruturada. Ele é utilizado quando o programa tem que comparar um
valor e a partir deste tomar uma decisão. Atenção, você sempre deverá finalizar o
comando IF com um ENDIF.

DO CASE ... CASE <CONDIÇÃO1> <COMANDOS> ... CASE <CONDIÇÃO2> <COMANDOS> ... ENDCASE

O comando CASE é bastante semelhante ao IF, a diferença é que ele é mais
simplificado.

DO WHILE <CONDIÇÃO> <COMANDOS> ENDDO

O comando DO WHILE é utilizado quando desejamos repetir alguns comandos um
número determinado de vezes. Atenção, a instrução ENDDO precisa encerrar a
estrutura de comando DO WHILE.

LOOP

O comando LOOP vai imediatamente para o início do DO WHILE ... ENDDO atual.
Qualquer comando ou série de comandos após o comando LOOP é ignorado.

EXIT

O comando EXIT interrompe a execução de comandos dentro de um comando DO
WHILE ... ENDDO, transferindo a execução para a instrução de comando imediatamente posterior ao comando ENDDO.

VALID(<VARIÁVEL$"CONDIÇÃO">)

Este comando é utilizado juntamento com o comando @ GET para testar a validade
de uma entrada de dados. É utilizado quando desejamos que o usuário digite uma
determinada tecla. Se esta tecla não for digitada o programa ficará esperando até o
usuário digitar a opção determinada em "CONDIÇÃO".

Ex.:

```
DO WHILE .T.
CLEAR SCREEN
```

```
BORDA = CHR(201) + CHR(205) + CHR(187) + CHR(186) + ;
CHR(188) + CHR(205) + CHR(200) + CHR(186) + ;
CHR(176)
@ 1,1,20,78 BOX BORDA
@ 03,03 TO 18,76
@ 4,4 CLEAR TO 17,75
VPRODUTO = SPACE(30)
VCUSTO = 0
VVENDA = 0
VLUCRO = 0
VFIM = SPACE(01)
@ 06,06 SAY "NOME DO PRODUTO: " GET VPRODUTO PICT "@!"
@ 08,06 SAY "PREÇO DE CUSTO: " GET VCUSTO PICT "99,999,999.99"
@ 10,06 SAY "PREÇO DE VENDA: " GET VVENDA PICT "99,999,999.99"
@ 12,06 SAY "LUCRO DA VENDA: "
READ
VLUCRO = VVENDA - VCUSTO
@ 12,23 SAY VLUCRO PICT "99,999,999.99"
@ 16,12 SAY "DESEJA CONTINUAR [S/N]" GET VFIM PICT "@!"
VALID(VFIM$"SN")
READ
IF VFIM="S"
LOOP
ELSE
EXIT
ENDIF
ENDDO
CLEAR
@ 12,35 SAY "ACABOU !!!!!!!"
```

Exercícios da Aula 03

Topo

1º - Escreva um programa que entre com as 3 notas de um aluno, calcule a média e mostre o resultado. Se a média for maior ou igual a 7 escrever "APROVADO" senão escrever "REPROVADO". Atenção, o programa deverá ficar repetindo até quando o usuário achar necessário.

2º - Escreva um programa que entre com o nome do funcionário, o salário bruto e o
percentual de aumento. Calcule o salário líquido. Se o salário líquido for maior que 5
salários mínimos escreva "SALÁRIO RAZOÁVEL" senão escreva "SALÁRIO
BAIXO". Atenção, o programa deverá ficar repetindo até quando o usuário achar
necessário.

Aula 04 - Comandos para a Criação de um Menu de Barras Luminosas

Topo
Agora que nós aprendemos a entrar dados, nós iremos aprender a criar um menu com
barras luminosas, muito utilizado para que o usuário escolha o que ele quer fazer.

```
@ L1,C1 PROMPT "ITEM DA BARRA" MESSAGE "QUALQUER MENSAGEM"
```

O comando PROMPT ... MESSAGE coloca seleções de menu na tela, destaca opções
de menu e permite que o cursor se mova através das teclas de cursor ou de entrada
direta. Atenção, são permitidos no máximo 32 PROMPTS por comando MENU.

```
MENU TO <VARIÁVEL>
```

O comando MENU TO permite a criação de prompts e de opções de menu mais
fáceis e mais eficientes. Este comando deve ser usado em conjunto com o comando
PROMPT e deve vir sempre abaixo do mesmo.

SET MESSATO TO <Nº DA LINHA>

Este comando é elaborado para funcionar com os comandos MENU TO e PROMPT.
Com o comando SET MESSAGE TO, selecione um número de linha entre 1 e 24 em
que a mensagem do PROMPT será exibida sempre que o cursor for colocado na
opção PROMPT.

SET WAP ON/OFF

O comando SET WRAP ativa e desativa a distribuição de MENU. Se estiver no seu
estado padrão: OFF sempre que se chegar a última posição do menu, só será possível
retornamos a primeira se voltarmos todas as posições.

Obs.: Você poderá enfeitar o menu usando os comandos para a elaboração de janelas
e o comando @ SAY para escrever a data atual na tela usando a função DATE(). Ex.:
@ 02,02 SAY DATE()

Escreverá a data atual na linha 2 e coluna 2.

Ex.:

```
OP = 1
DO WHILE .T.
CLEAR SCREEN
SET MESSAGE TO 23
SET WRAP ON
@ 09,17 PROMPT " 1.INCLUSÃO " MESSAGE " INCLUSÃO DE DADOS"
@ 10,17 PROMPT " 2.CONSULTA " MESSAGE " CONSULTA DE DADOS"
@ 11,17 PROMPT " 3.EXCLUSÃO " MESSAGE " EXCLUSÃO DE DADOS"
@ 12,17 PROMPT " 4.FINALIZA " MESSAGE " FINAL DE OPERAÇÃO"
```

```
MENU TO OP
dO CASE
CASE OP=4
CLEAR SCREEN
QUIT
ENDCASE
ENDDO
```

Comandos para gravar toda a tela numa variável e depois recuperar a tela.

SAVE SCREEN TO <VARIÁVEL>

Grava toda a tela na variável correspondente.

RESTORE SCREEN FROM <VARIÁVEL>

Recupera a tela que foi gravada na variável.

Exercícios da Aula 04

Topo
1º - Escreva um programa que monte um MENU que tenha 6 opções:

1.Cadastramento, 2. Consulta, 3. Alteração, 4. Exclusão, 5. Relatórios, 6. Final
do programa.

Atenção, o MENU deverá estar cercado por uma janela de barras duplas e as
mensagens do PROMPT deverão estar na linha de número 22.

2º - Escreva um programa para quando for selecionada a opção 5 do menu anterior,

apareça na tela outro menu com as opções: 1. Ordem de código, 2. Ordem alfabética,
3. Retorna ao menu.

Aula 05 - Programa Usando Arquivo

Topo
Até agora todas as informações que nós pedimos ao usuário ficaram na memória, de
forma que quando o usuário desligar o computador todas as informações serão
perdidas, portanto nós iremos tratar de gravar essas informações no disco rígido. E
para fazer isso nós usaremos um programa chamado DBASE e dentro dele usar o
comando CREAT NOMEARQ. Por exemplo, vamos imaginar que iremos fazer uma
agenda eletrônica e que iremos guardar o código, o dia, a hora e o compromisso.

Para fazer isso:

1º - CD\DBASE <ENTER>

2º - DBASE <ENTER>

3º - Usar o comando CREAT: .CREAT AGENDA <ENTER>

Daí aparece uma tela nova pedindo para se digitar os campos:

Field Name Type Width Dec

CODIGO N 03 0
DIA C 10

HORA C 05
COMPRO C 30

4º - Com o arquivo criado podemos sair do DBASE: . QUIT <ENTER>

Comandos Utilizados para Gravar os Dados no Arquivo

Topo

APPEND BLANK

Este comando deve ser utilizado quando desejamos acrescentar mais alguma
informação no arquivo de dados.

REPLACE <VARIÁVEL DO ARQUIVO> WITH <VARIÁVEL DA MEMÓRIA>

Este comando transfere o valor da variável de memória para a variável do arquivo.

SELECT <Nº DO ARQUIVO>

Este comando serve para determinarmos quantos e quais são os arquivos que iremos
usar em nosso programa.

INDEX ON <VARIÁVEL> TO <ARQUIVO>

Coloca o arquivo em ordem alfabética pelo campo determinado pela variável.

Atenção: Sempre que um programa for utilizar arquivo, você deverá determinar no
menu principal os arquivos que serão utilizados e como eles serão organizados.

Ex.:

```
SELECT 1
USE AGENDA
IF .NOT. FILE("INDCOD.NTX")
INDEX ON CODIGO TO INDCOD
```

```
ELSE
SET INDEX TO INDCOD
ENDIF
USE AGENDA INDEX INDCOD
```

Vamos agora montar o programa que gravará as informações no arquivo agenda:

```
SELECT 1
SET ORDER TO 1
@ 01,01 TO 20,78 DOUBLE
DO WHILE .T.
VCODIGO = 0
VDIA = SPACE(10)
VHORA = SPACE(05)
VCOMPRO = SPACE(30)
VFIM = SPACE(01)
@ 05,06 SAY "CODIGO: " GET VCODIGO PICT "999"
READ
SEEK VCODIGO
IF FOUND()
@ 22,32 SAY "JÁ CADASTRADO"
INKEY(0)
@ 22,32 SAY "                                "
LOOP
ENDIF
@ 07,6 SAY "DIA: " GET VDIA PICT "@!"
@ 09,6 SAY "HORA: " GET VHORA PICT "@!"
@ 11,6 SAY "COMPROMISSO: " GET VCOMPRO PICT "@!"
@ 15,48 SAY "CONTINUAR? [S/N] " GET VFIM PICT "@!"
VALID(VFIM$"SN")
READ
APPEND BLANK
REPLACE CODIGO WITH VCODIGO
REPLACE DIA WITH VDIA
REPLACE HORA WITH VHORA
REPLACE COMPRO WITH VCOMPRO
IF VFIM = "S"
LOOP
ELSE
EXIT
ENDIF
ENDDO
```

Trabalho para Casa

Topo
Escreva em uma folha 2 programas. O 1º será um menu com as seguintes opções: 1.
Inclui, 2. Consulta, 3. Altera, 4. Exclui, 5. Fim.

Quando for escolhida a opção nº 1 o 2º programa deverá guardar em um arquivo o
código, cliente, endereço, bairro, cidade, cep, telefone e data de nascimento. Este 2º
programa deverá ficar repetindo até que o usuário queira encerrar.

Atenção: Você deverá colocar todos os passos para a criação do arquivo de dados.

Aula 06 - Consulta usando o comando SEEK

Topo

SEEK <VARIÁVEL>

O comando SEEK é utilizado quando desejamos encontrar em um banco de dados um
determinado registro.

FOUND()

Esta função funciona junto com o comando SEEK. Se for encontrado o registro a
função é tida como verdadeira caso contrário será falsa.

Exemplo de um módulo para consulta:

Atenção: Usaremos o banco de dados Agenda.

```
SELECT 1
SET ORDER TO 1
@ 01,01 TO 20,78 DOUBLE
DO WHILE .T.
VCODIGO = 0
VFIM = SPACE(01)
@ 05,06 SAY "CODIGO: " GET VCODIGO PICT "999"
```

```
READ
SEEK VCODIGO
IF .NOT. FOUND()
@ 22,32 SAY "ESTE CÓDIGO NÃO FOI CADASTRADO"
INKEY(0)
@ 22,32 SAY " "
LOOP
ENDIF
@ 07,6 SAY "DIA: " + DIA
@ 09,6 SAY "HORA: " + HORA
@ 11,6 SAY "COMPROMISSO: " + COMPRO
@ 15,48 SAY "CONTINUAR? [S/N] " GET VFIM PICT "@!"
VALID(VFIM$"SN")
READ
IF VFIM = "S"
LOOP
ELSE
EXIT
ENDIF
ENDDO
```

Exercício da Aula 06

Topo

1º - Escreva o módulo para consulta do trabalho que foi feito na aula anterior.

Aula 07 - Alteração usando o comando SEEK

Topo

Agora nos vamos criar um programa que faça a alteração dos dados já incluídos. Ele
irá dar um GET diretamente nas variáveis do arquivo.

```
SELECT 1
SET ORDER TO 1
@ 01,01 TO 20,78 DOUBLE
```

```
DO WHILE .T.
VCODIGO = 0
VFIM = SPACE(01)
@ 05,06 SAY "CODIGO: " GET VCODIGO PICT "999"
READ
SEEK VCODIGO
IF .NOT. FOUND()
@ 22,32 SAY "ESTE CÓDIGO NÃO FOI CADASTRADO"
INKEY(0)
@ 22,32 SAY "                                                    "
LOOP
ENDIF
@ 07,6 SAY "DIA: " GET DIA PICT "@!"
@ 09,6 SAY "HORA: " GET HORA PICT "@!"
@ 11,6 SAY "COMPROMISSO: " GET COMPRO PICT "@!"
@ 15,48 SAY "CONTINUAR? [S/N] " GET VFIM PICT "@!"
VALID(VFIM$"SN")
READ
IF VFIM = "S"
LOOP
ELSE
EXIT
ENDIF
ENDDO
```

Exercício da Aula 07

Topo

1º - Escreva o módulo para alteração do trabalho que foi feito na aula 05.

Aula 08 - Exclusão usando o comando SEEK

Topo

O que acontece se nós quisermos apagar alguma ficha? Essa "seção" faz isso, ela
exclui uma ficha depois de perguntar ao usuário se ele quer mesmo isso.

```
SELECT 1
SET ORDER TO 1
@ 01,01 TO 20,78 DOUBLE
DO WHILE .T.
VCODIGO = 0
VFIM = SPACE(01)
@ 05,06 SAY "CODIGO: " GET VCODIGO PICT "999"
READ
SEEK VCODIGO
IF .NOT. FOUND()
@ 22,32 SAY "ESTE CÓDIGO NÃO FOI CADASTRADO"
INKEY(0)
@ 22,32 SAY " "
LOOP
ENDIF
@ 07,6 SAY "DIA: " + DIA
@ 09,6 SAY "HORA: " + HORA
@ 11,6 SAY "COMPROMISSO: " + COMPRO
@ 15,48 SAY "DESEJA EXCLUIR? [S/N] " GET VFIM PICT "@!"
VALID(VFIM$"SN")
READ
IF VFIM = "S"
DELETE
PACK
LOOP
ELSE
EXIT
ENDIF
ENDDO
```

Neste módulo os únicos comandos novos que apareceram foram DELETE e PACK.
O comando DELETE marca um determinado registro do banco de dados e o comando
PACK elimina o registro marcado.

Exercício da Aula 08

Topo
1º - Escreva o módulo para alteração do trabalho que foi feito na AULA 05.

Aula 09 - Como Elaborar Relatórios
Usando o Clipper?

Topo

DO WHILE .NOT. EOF()

Este comando significa o seguinte: Enquanto não for encontrado o final do arquivo
faça ou enquanto existir alguma informação no arquivo faça. Tudo o que vier abaixo
deste comando ficará repetindo dependendo do número de informações que
existirem dentro do arquivo.

SKIP

Este comando passa para a próxima informação existente no arquivo de dados.

ISPRINTER()

Esta função verifica se a impressora está pronta para impressão.

PROW()

Esta função retorna a posição atual de linha da cabeça de impressão da impressora.

PCOL()

Esta função retorna a posição atual de coluna da cabeça de impressão da impressora.

SET DEVICE TO PRINTER

Tudo o que for escrito com o comando @ SAY sairá na impressora. Para voltar a
escrever na tela use o comando SET DEVICE TO SCREEN. Atenção, sempre no final

do relatório, você deverá usar o comando SET DEVICE TO SCREEN.

Ex.: Vamos imaginar que iremos fazer um relatório para o programa agenda onde
queremos que saia no relatório o dia, a hora e o compromisso.

```
SAVE SCREEN TO TELA18
DO WHILE .T.
IMPPRO = SPACE(01)
@ 12,24 CLEAR TO 16,54
@ 13.28 SAY "IMPRESSORA PRONTA? [S/N] " GET IMPPRO PICT "@!" +;
VALID(IMPPRO$"SN")
READ
IF IMPPRO = "N"
RESTORE SCREEN FROM TELA18
RETURN
ELSE
SET CURSOR OFF
SET COLOR TO W+/G+
IF .NOT. ISPRINTER()
SET COLOR TO W+/G
@ 16,26 SAY "IMPRESSORA NÃO ESTÁ PRONTA! "
INKEY(0)
@ 16,26 SAY "                                                    "
LOOP
ENDIF
EXIT
ENDIF
ENDDO
SET COLOR TO W+/G
@ 12,24 CLEAR TO 16,54
@ 13,28 SAY "I M P R I M I N D O "
SET DEVICE TO PRINTER
** EFETUAR IMPRESSÃO DO ARQUIVO ORGANIZADO
LIN = 0
PAG = 1
HO = TIME()
DA = DTOC(DATE())
TC = 0
SELECT 1
SET ORDER TO 1
GOTO TOP
TOC = RECCOUNT()
DO WHILE .NOT. EOF()
```

```
@ PROW() + 00, 00 SAY "TESTE DE RELATÓRIO"
@ PROW() + 01, 02 SAY "TOTAL DE REGISTROS: " +
ALLTRIM(STR(TOC))
@ PROW() + 00, 23 SAY "DATA: "+DA + " HORA: " + HO
@ PROW() + 00, 72 SAY "PAG.: " + ALLTRIM(STR(PAG,0))
@ PROW() + 01, 00 SAY REPLICATE("-",79)
DO WHILE .NOT. EOF()
@ PROW() + 01, 02 SAY "DIA: " + DIA
@ PROW() + 01, 02 SAY "HORA: " + HORA
@ PROW() + 01, 02 SAY "COMPROMISSO: " + COMPRO
@ PROW() + 01, 00 SAY REPLICATE("=",79)
TC = TC + 1
IF LIN > 55
LIN = 0
PAG = PAG + 1
EJECT
EXIT
ENDIF
ENDDO
ENDDO
SET DEVICE TO SCREEN
RESTORE SCREEN FROM TELA18
RETURN
```

Exercício da Aula 09

Topo

1º - Escreva o módulo para o relatório do trabalho que foi feito na Aula 05.

Aula 10 - Uma Consulta Profissional
Usando a Função DBEDIT()

Topo

A função DBEDIT() é bastante poderosa. Com ela você consegue pesquisar qualquer
informação do arquivo de dados com uma rapidez impressionante. Veja no exemplo

abaixo como utilizar esta poderosa função. Vamos imaginar uma consulta para o
programa AGENDA.

```
SET COLOR TO
SAVE SCREEN TO TELA5
CLEAR SCREEN
* DECLARA 3 VETORES COM O Nº DE INFORMAÇÕES A SEREM PESQUISADAS.
PUBLIC VETOR1[04],VETOR3[04],VETOR4[4]
SET SCOREBOARD OFF
*PREENCHE VETOR COM NOME DOS CAMPOS
VETOR1[1] = "CODIGO"
VETOR1[2] = "DIA"
VETOR1[3] = "HORA"
VETOR1[4] = "COMPRO"
*CABEÇALHOS
VETOR3[1] = "CODIGO: "
VETOR3[2] = "DIA DO COMPROMISSO: "
VETOR3[3] = "HORÁRIO: "
VETOR3[4] = "COMPROMISSO: "
*MÁSCARAS
VETOR4[1] = "999"
VETOR4[2] = "@!"
VETOR4[3] = "99:99"
VETOR4[4] = "@!"
SELECT 1
SET ORDER TO 1
GOTO TOP
SET COLOR TO I
@ 23,00 SAY " [ESC] FIM "
SET COLOR TO W/G+
@ 0,0 TO 3,79
@ 1,1 CLEAR TO 2,78
@ 1,2 SAY "DATA ATUAL "
@ 2,3 SAY DATE()
@ 1,28 SAY " LISTAGEM GERAL "
@ 2,28 SAY " XLK INFORMÁTICA "
@ 1,60 SAY "VERSÃO DO SISTEMA "
@ 2,60 SAY " 2.0 "
@ SET COLOR TO W/B+
@ 4,0 TO 22,79 DOUBLE
*CHAMA A FUNÇÃO
DBEDIT(5,1,22,78,VETOR1,.T.,VETOR4,VETOR3,.T.,.T.,.T.)
```

```
SET COLOR TO
SET CURSOR OFF
RESTORE SCREEN FROM TELA5
RETURN
```

Exercícios da AULA 10

Topo

1º Escreva o módulo para consulta usando DBEDIT() do trabalho que foi feito na
AULA05.

Teste de Lógica

Topo

1 - Escreva um programa para escrever na tela a relação de todos os números pares
de 1 a 1000.

2 - Escreva um programa que calcule a soma de todos os números de 1 a 1000.

Funções

Topo
Funções são também pedaços de programas que facilitam o trabalho do programador.

O que uma FUNÇÃO faz é retornar um valor ao programa.

Ex.: Escrever uma função que retorne o quadrado de um número qualquer.

```
* FUNÇÃO PARA RETORNAR O QUADRADO DE UM NÚMERO
* PARÂMETROS - NUMERO
FUNCTION QUADRADO
PARAMETER NUMERO
QUADRADO=(NUMERO*NUMERO)
RETURN(QUADRADO)
```

Ex.: Escrever uma função que retorne o valor de um número elevado a uma determina
potencia.

```
* FUNÇÃO PARA CALCULAR A EXPONENCIAÇÃO. EX.: 23.
* PARAMETROS - NUMERO,EXPOENTE
FUNCTION POTENCIA
PARAMETER NUMERO,EXPOENTE
VALOR = NUMERO
FOR REPETE=1 TO EXPOENTE-1
VALOR=VALOR*NUMERO
NEXT
RETURN(VALOR)
```

Como Utilizar uma FUNÇÃO?

Topo
Você deverá escrever suas funções em um arquivo separado. Para isso, você deverá
fazer:

1º) Crie um arquivo com um nome qualquer.

Ex.: FUNCAO.PRG

2º) Digite as funções exemplos conforme a apostila.

3º) Compile esse arquivo para verificar se não há erros. E também para não ser
necessário compilá-lo toda vez que for compilar o programa principal.

4º) Crie um outro arquivo com um nome qualquer.

Ex.: TESTE2.PRG

5º) Na primeira linha deste arquivo coloque o seguinte comando: SET PROCEDURE
TO FUNCAO

Agora você já pode utilizar qualquer uma daquelas funções como se fosse um
comando do CLIPPER. Por exemplo, digite o programa abaixo:

```
SET PROCEDURE TO FUNCAO
CLEAR SCREEN
@ 1,1 TO 20,78 DOUBLE
@ 5,5 SAY "O VALOR DE 8 AO QUADRADO: "
@ 5,31 SAY QUADRADO(8)
@ 9,5 SAY "O VALOR DE 8 AO CUBO: "
@ 9,31 SAY POTENCIA(8,3)
INKEY(0)
QUIT
```

Outros Comandos

Topo

MODIFY STRUCTURE

Modifica a estrutura do arquivo de dados do tipo DBF.

DELETE

Permite marcar registros para remoção.

PACK

Elimina os registros marcados.

ZAP

Destrói todos os dados de um arquivo DBF.

RECNO()

Retorna o número de registros existentes em um arquivo de dados.

QUIT

Sai do programa imediatamente.

SET COLOR TO

Este comando altera as cores exibidas na tela.

COR	NÚMERO	LETRA
PRETO	0	N
AZUL	1	B
VERDE	2	G
CIANO	3	BG
VERMELHO	4	R
MAGENTA	5	RB
MARRON	6	GR
BRANCO	7	

		W
CINZA		
	8	
		N+
AMARELO		
	9	
		GR+
INVERSO		
		I
VAZIO		
		X

DICA: Para obter uma cor que apareça mais forte utilize a letra correspondente seguida do sinal de +. Ex: W+ branco forte

Para obter uma cor que apareça piscando, utilize a letra correspondente seguida pelo sinal de *. Ex: W* branco piscante.

Você pode ainda utilizar os 2 recursos juntos. Ex: W+* branco forte e piscante.

SET CURSOR ON/OFF

Ativa ou desativa a exibição do cursor.

LASTKEY()

Retorna a última tecla que foi digitada.

Comandos de Configuração

Topo

SET TALK OFF

Usado para configurar o vídeo

SET SCOREBORARD OFF

Usado para eliminar as mensagens das teclas Ins, Caps e Num.

SET DATE FRENCH

Usado para mudar a data do formato MM/DD/AA para DD/MM/AA

Comandos de Data e Hora

Topo

DATE()

Retorna a data no formato especificado por SET DATE

TIME()

Retorna a hora do sistema operacional.