Clipper (linguagem de programação)

Origem: Wikipédia, a enciclopédia livre.
(Redirecionado de **CA-Clipper**)

Clipper (ou **CA-Clipper**) é um compilador 16 bits da linguagem xBase para o ambiente DOS. Foi criada em 1984 com o propósito de ser um compilador para o Ashton-Tate dBase, um gerenciador de banco de dados muito popular em sua época.

Trata-se de uma derivação da Clipper Summer, e após ser adquirida pela Computer Associates chegou á versão 5.3B, implementada por uma inteface gráfica compatível com o MS-Windows 3.11 e por um subconjunto de suporte para as linguagens C e Assembler, o que tornou possível um protótipo de orientação a objetos.

Quando a Computer Associates parou de oferecer suporte a essa linguagem, ela era destinada ao desenvolvimento de aplicações para as plataformas MS-DOS e oferecia bibliotecas para suporte de rede.

Por utilizar-se de um padrão de linguagem originalmente desenvolvido pela Ashton Tate e existente desde os sistemas operacionais CP/M, possui uma sintaxe bem distinta das linguagens mais atuais. Em sua época, era considerada uma linguagem intuitiva e elegante, utilizando-se de pequenos verbos e abreviações, símbolos e estruturação.

Seus compiladores geravam executáveis que em 2008 seriam considerados minúsculos, extremamente rápidos e, para a maioria dos usuários, com uma interface pouco amigavél. Novamente, em sua época, era uma das mais versáteis e possibilitava a criação de sistemas totalmente integrados com imagens, sons e vídeo, e já se utilizava dos conceitos de hyperlink (via padrão rtf de texto), ajuda de contexto e instanciamento de objetos (ainda que primitivos, via Code Blocks), coisa que, exceto pela linguagem C, só foi alcançada anos mais tarde pela concorrência.

Outra característica importante foi a inclusão da tecnologia Rushmore, hoje pertencente a Microsoft, de indexação para suas tabelas de dados, tornando-a uma das linguagens de melhor desempenho nessa área.

Mas sistemas originais criados com essa linguagem requerem ajustes constantes para se tornar utilizáveis em sistemas operacionais mais modernos. E como não há mais suporte oficial para ela, grupos de usuários e desenvolvedores resolvem os problemas que vão surgindo com a constante evolução da informática por si mesmos, mediante bibliotecas OpenSource, patches, portings e outras criativas soluções.

Mais algumas características da linguagem:

- Pré-processador de código-fonte;
- Compiladores de alta-performance;
- Depurador interativo;
- IDE gráfica (opcional, requerendo o MS-Windows® instalado);
- Suporte a modos gráficos de vídeo VGA (com os drivers adequados);
- Suporte a mouse (com o driver do fabricante) integrado á bibliotecas de entradas de dados;
- Geração de executáveis que utilizavam os modos protegido ou real de memória (escolhendo-se um dos compiladores específicos para essas características);
- Geração de módulos de Overlay (grosso modo, equivalentes às bibliotecas de vínculo dinâmico), diminuindo o tamanho dos executáveis e seu uso de memória;
- Dois objetos reais para MS-DOS, (TBrowse e Get) para desenvolvimento de telas com massas de dados e de entradas de dados respectivamente;
- Teclas de aceleração (o equivalente às teclas de atalho);

O conjunto do objeto Get disponibilizava ao desenvolvedor itens de interface tais como check- box, listbox, botões de opção, botões de rádio, barras de rolagem, barras de menus e itens de menu, dentre outros. Esses itens eram visíveis em MS-DOS mediante caracteres semi-gráficos da tabela ASCII estendida.

A versão 5.03 sofreu uma atualização radical, tendo sido desenvolvido com o uso da linguagem C da Microsoft e, com isso, tornando possível a utilização de processadores aritméticos, se estivessem presentes no computador. Com isso, aplicações eram executadas com até 30% á mais de desempenho, sem qualquer alteração no código-fonte.

Índice

- 1 História
- 2 Utilização Decrescente
- 3 Extensão dos Arquivos Manipulados Pelo Clipper
- 4 Informações Complementares
- 5 Exemplo de Compilador
- 6 Exemplos de sintaxe
- 7 Exemplos de Comandos Básicos e Laços de Repetição
- 8 Exemplos de código
- 9 Ver também
- 10 Ligações externas

História

Conta a lenda que dois amigos estavam almoçando num restaurante de frutos do mar chamado Nantucket Lighthouse, discutindo como era frustrante o fato da Ashton-Tate se recusar a criar um compilador para o seu principal produto. A baixa velocidade de processamento do dBase quando comparado às aplicações compiladas era gritante. Começaram então a discutir a idéia de criar um compilador e fundar uma empresa para comercializá-lo. O nome Clipper veio de um quadro na parede do restaurante que mostrava um destes rápidos e elegantes navios mercantes. O nome da empresa foi uma escolha emprestado do nome do restaurante.

Quando de sua criação, o Nantucket Clipper se propunha basicamente a ser o melhor compilador para dBase que existia. Na caixa da versão Summer '87 vinha os dizeres "dBase III ® Compiler". As versões foram:

Nantucket Corporation; com nome de estações do ano, vendidas como "dBase compilers"

- Nantucket Clipper Winter '84 - released May 25 1985
- Nantucket Clipper Summer '85 - released 1985
- Nantucket Clipper Winter '85 - released January 291986
- Nantucket Clipper Autumn '86 - released October 31 1986
- Nantucket Clipper Summer '87 - released December 21 1987

Nantucket Corporation; Clipper 5

- Nantucket Clipper 5.00 - released 1990
- Nantucket Clipper 5.01 - released April 15 1991
- Nantucket Clipper 5.01 Rev.129 - released March 31 1992

Computer Associates

- CA-Clipper 5.01a -
- CA-Clipper 5.20 - released February 15 1993
- CA-Clipper 5.2a - released March 15 1993
- CA-Clipper 5.2b - released June 25 1993
- CA-Clipper 5.2c - released August 6 1993
- CA-Clipper 5.2d - released March 25 1994
- CA-Clipper 5.2e - released February 7 1995
- CA-Clipper 5.30 - released June 26 1995
- CA-Clipper 5.3a - released May 20 1996
- CA-Clipper 5.3b - released May 20 1997

Com a versão 5, Clipper iniciou o processo de desvinculação do dBase, tornando-se uma linguagem de programação com linha de evolução própria. A versão 5 adicionou recursos inexistentes no dBase, incluindo um depurador visual, exceções estruturadas, RDD (um tipo de ODBC pré-histórico), novos tipos, novas funções para gerenciar arrays e um pré-processador que permitiu que a linguagem pudesse ser estendida de forma praticamente ilimitada no ambiente de 16 bits.

Antes de a Computer Associates comprar a Nantucket, o escritório alemão da Nantucket havia começado um projeto de maneira informal conhecido como "ASPEN", embora internamente fosse chamado simplesmente de Clipper for Windows. O projeto representava uma ruptura com as versões anteriores do Clipper na medida em que introduzia o conceito de orientação ao objeto (**OO**) e suporte ao ambiente gráfico da **Microsoft** sem se preocupar com retro-compatibilidade. Dentre os novos recursos, estava um desempenho comparável à do C++ com uma linguagem muito mais acessível e que possuía uma enorme base de potenciais programadores advindos do dBase, Clipper e outros ambientes **XBASE**.

A CA resolveu que iria entrar para valer na briga por uma fatia considerável do nascente mercado de programação para Windows e adquiriu a Nantucket por causa do VO (Visual Objects), apostando alto no que foi um dos maiores fracassos da área de tecnologia. A falta de compatibilidade com as versões anteriores levou muitos desenvolvedores a migrarem para a nova ferramenta da **Borland** chamada **Delphi**.

O Clipper, que após a aquisição foi renomeado CA-Clipper, ainda viu uma edição que usava o gerenciamento de memória do Windows enquanto mantinha a interface caracter, mas foi oficialmente aposentado em favor do novo produto, o VO.

Embora hoje seja considerada uma linguagem obsoleta dado que parou de evoluir após a versão 5.3 e tratando-se o VO de um produto distinto e nati-morto, Clipper ainda possui uma razoável base de programadores conhecidos pelo depreciativo apelido de "clippeiros". Projetos open-source como o Projeto Harbour continuam a oferecer suporte ao padrão **XBASE** enquanto são orientados para modernos ambientes gráficos, embora sem nenhum apoio oficial da CA, detentora dos direitos sobre o Clipper.

Em resumo, o Clipper permite a dinamização de aplicações com arquivos de dados, tornando-as mais fáceis e rápidas que as desenvolvidas em uma linguagem de programação tradicional, como Cobol, Basic ou Pascal. Com uma simples, moderna e eficiente linguagem de programação, permite o encadeamento ordenado e lógico de seus comandos possibilitando rapidamente a definição de programas com alto grau de complexidade e sofisticação, permitindo inclusive interações com outras linguagens como “C” e Assembler, que lhe confere a flexibilidade necessária para a utilização profissional.

SUMMER 85 - nesta versão, o CLIPPER era totalmente compatível com a versão 1.0 do DBASE III, chegando a ser bem próximo dele, porém, apresentando alguns recursos adicionais como:

- Maior capacidade de manipulação de arquivos e variáveis; - Construção de "HELP" ao usuário; - Múltiplo relacionamento entre arquivos; - Criação de funções-de-usuário (UDF's); - Novos comandos e funções que não existiam no DBASE III.

WINTER 85 - ainda permanecia a compatibilidade com o DBASE III entretanto surgiram algumas implementações, dentre elas, as principais são:

- Variáveis indexadas: vetores; - Surgimento do comando @...PROMPT (menu de barras); - Novas funções para manipulação de campos MEMO. Logo após a versão WINTER 85 do CLIPPER foi lançado em contra-ataque, o DBASE III PLUS que incluía os vários comandos e funções que o CLIPPER já possuía só que com a principal novidade, que era a possibilidade de trabalho em ambiente de rede local. Para acompanhar a versão do DBASE, foi lançada a versão:

AUTUMN 86 - nesta versão, o CLIPPER também passou a trabalhar em ambiente de rede local, ganhou novos comandos e funções, mas muitos dos novos recursos do DBASE III PLUS foram implementados de forma provisória, através de rotinas auxiliares, escritas em linguagem C e Assembler. A compatibilidade com o DBASE III PLUS ainda existia.

SUMMER 87 - nesta versão ocorre dois fatores importantes para o estágio de desenvolvimento do CLIPPER, são eles:

- Mudança do compilador C, através do qual era construído. - Decisão de se separar de uma vez do DBASE. Com isso o CLIPPER se transformou numa ferramenta realmente destinada à construção de sistemas profissionais. Além de uma quantidade considerável de novos comandos e funções, a arquitetura do CLIPPER foi praticamente aberta. Tornou-se possível escrever uma função qualquer utilizando o Microsoft C e linkeditá-la diretamente com as bibliotecas e módulos objeto gerados pelo CLIPPER.

VERSÃO 5.0 - a tendência já observada na versão SUMMER 87 confirmou-se. A compatibilidade com a linguagem DBASE, apesar de mantida, tornou-se apenas uma circunstância histórica. A nova estrutura de programação do CLIPPER e os novos e sofisticados recursos baseados na estrutura de programação da linguagem C e tendências de programação orientada a objetos, indicam um afastamento definitivo do padrão DBASE.

NOVIDADES DA VERSÃO 5.0 - Dentre as várias novidades que a versão 5.0 do CLIPPER nos trouxe, destacamos algumas logo abaixo:

- Acesso ao pré-processador do compilador (diretivas); - Novo compilador com recursos e opções mais otimizadas; - Novo linkeditor (RTlink), que permite a criação de overlays dinâmicos; - Help "on-line" para programador (Norton Guide); - Novos operadores; - Definições de funções-de-usuário (UDF's); - Debugador mais eficiente; - Novos tipos e classes de variáveis; - Matrizes multidimensionais, etc.

Utilização Decrescente

Tanto o **dBase** quanto o **Clipper** são produtos de uma época onde os computadores pessoais eram desconectados, e o banco de dados era um conjunto de arquivos em disco acessado por apenas um usuário. Ambos os programas funcionam, na prática, como uma biblioteca ligada ao programa final, monolítico, que acessa diretamente os arquivos contendo os dados, sem intermediação (como ocorre no caso dos SGDB).

Com o aparecimento das redes de computador, passou a ser possível utilizar discos compartilhados para acessar diretamente esses arquivos, porém fazendo que o programador tivesse que controlar e resolver vários problemas ligados ao acesso compartilhado de arquivos e registros.

Atualmente, apesar de muitos programas ainda utilizarem essas linguagens, o uso de um SGBD é mais recomendado, o que leva, gradativamente, ao abandono dessa tecnologia.

Para o Clipper foram desenvolvidos RDDs que permitem o uso de SGBD, como o ADS Advantage Database Server, o RaSQL/b para Btrieve/Pervasive e o UltiRoute, que possibilita o acesso a qualquer SGBD via ODBC.

Extensão dos Arquivos Manipulados Pelo Clipper

O CLIPPER como qualquer outra linguagem de programação possui os seus próprios arquivos e extensões para que sejam facilmente reconhecidos por um programador. Logo abaixo, serão discriminados os vários arquivos manipulados pelo CLIPPER.

- .PRG = arquivos de programa-fonte
- .CH = arquivos-cabeçalho ou arquivos include
- .OBJ = arquivos-objeto
- .LIB = arquivos de bibliotecas
- .TMP = arquivos temporários
- .PPO = arquivos do pré-processador
- .EXE = arquivos auto-executáveis
- .DBF = arquivos de dados
- .DBT = arquivos de campo memo
- .NTX = arquivos de índices
- .MEM = arquivos de variáveis de memória
- .LBL = arquivos de definição de etiquetas
- .FRM = arquivos de definição de relatórios
- .FMT = arquivos de formatação
- .CLP = arquivos script ou lista de clippagem
- .LNK = arquivos de linkedição
- .PLL e PLT = arquivos de biblioteca pré-linkadas
- .OVL = arquivos de overlay
- .MAP = arquivos de alocação de memória

Informações Complementares

Com o Clipper é possível:

• Criar, organizar, classificar, copiar, selecionar e relacionar conjuntos de arquivos que formam o Banco de

Dados;

• Adicionar, alterar, eliminar, exibir e listar global ou seletivamente as informações contidas nos arquivos de dados;

• Gerar relatórios padronizados, efetuar automaticamente somas, agregações, contagens e operações aritméticas sobre os valores dos dados armazenados nos arquivos;

• Formatar telas de entrada de dados no vídeo e gerar relatórios, tabelas e listagens complexas na impressora, de acordo com as necessidades do usuário;

• Produzir Sistemas de Informação completos e integrados, com recursos e sofisticações encontrados apenas nos mais modernos softwares que hoje disputam o fabuloso mercado da microinformática.

Em resumo, o Clipper permite a dinamização de aplicações com arquivos de dados, tornando-as mais fáceis e rápidas que as desenvolvidas em uma linguagem de programação tradicional, como Cobol, Basic ou Pascal. Com uma simples e eficiente linguagem de programação, permite o encadeamento ordenado e lógico de seus comandos possibilitando rapidamente a definição de programas com alto grau de complexidade e sofisticação, permitindo inclusive interações com outras linguagens como “C” e Assembler, que lhe confere a flexibilidade necessária para a utilização profissional.

Paradigma Procedural: Clipper é pertencente ao Paradigma procedural (como Pascal, C, Ada, Cobol, Fortran, Clipper). Linguagens procedurais são aquelas no qual nosso código é dividido em subrotinas (procedures) ou function (funções). Uma procedure é uma função sem retorno (pode ser visto como uma função que não retorna nada, void). Se você realmente quiser modularizar seu programa, o melhor que você pode fazer é dividi-lo em várias subrotinas/funções, e num nível maior, em várias bibliotecas. Antes da programação estruturada, usava-se nos códigos o recurso do GOTO, que tornava a maior parte dos programas ilegíveis. Os famosos códigos espaguete.

Sistema de Tipos Utilizado pela Linguagem: Clipper possue uma tipagem dinâmica, porém surpreendentemente forte.O seguinte trecho de código passa pelo compilador, mas dá erro na execução:

```
local a,b

a = 1

b = 2

b = "x"

? a + b
```

O Clipper 5 introduziu a diretiva "local", que permite declarar as variáveis no início da função ou procedimento. É um passo em direção à tipagem estática; o plano da Nantucket era tornar o Clipper mais parecido com C ao longo de sua evolução. A comparação com linguagens de tipagem dinâmica, que resolvem nomes em run-time, é verdadeiramente covarde; mas por completeza:

```
a = tabela->campo // Clipper

$a = $linha->campo; // PHP

a = linha.campo ## Python
```

O Clipper suporta nativamente a sintaxe acima para tabelas DBF, mas não me permite definir um novo tipo

com essa sintaxe. Já PHP e Python permitem criar tipos com atributos dinâmicos, confortáveis ao usuário.

Exemplo de Compilador

O Harbour é um compilador de software livre para a linguagem Clipper (a linguagem que é implementada pelo compilador CA-Clipper). O Harbour é um compilador multi-plataforma e sabe-se que compila e executa em MS-DOS, MS-Windows, OS/2 e GNU/Linux. A principal diferença do Harbour para outros compiladores Dbase é que ele é um software livre.

Exemplos de sintaxe

- '**NUMÉRICO:** Aceita apenas números com ou sem casas decimais (decimals). O ponto da casa decimal conta no tamanho (width).

Exemplo: 999.99 (width 6, decimals 2)

- **CARACTER:**

Aceita letras e números. Tamanho máximo de 256 caracteres.

- **DATA:** Aceita apenas datas.
- **LÓGICO:** Aceita apenas Verdadeiro (.T.) ou Falso (.F.)
- **MEMO:** Aceita letras e números, é como um tipo de campo Caracter com tamanho de até 64Kb. Usado geralmente para armazenar "Observação" ou outras informações que precisem ser detalhadas. Ao adicionar um campo Memo no seu DBF, é criado um espelho dele com a extensão .DBT. (Se o RDD usado for o nativo do Clipper DBFNTX). Só é possível editar um campo memo com a função MEMOEDIT().

Exemplo: O DBF "clientes" tem um campo memo para observações dos clientes, então irá existir:

CLIENTES.DBF

CLIENTES.DBT

- **NUMÉRICO:** Aceita apenas números com ou sem casas decimais (decimals). O ponto da casa decimal conta no tamanho (width).

Exemplo:

nTOTAL = 0

nCAMPO = 125

- **CARACTER:** Aceita letras e números. Tamanho máximo de 256 caracteres.

Exemplo:

cNOME = SPACE(35)

cCAMPO = "ANDERSON"

- **DATA:** Aceita apenas datas. Considere o uso do comando SET DATE BRITISH para definir o padrão de data igual ao brasileiro dd/mm/aa. Considere também o uso do comando SET EPOCH TO 1980

para resolver o "Bug do Milênio", veja mais em Know-how - Y2K Bug do Ano 2000.

Exemplo:

dDATA = CTOD("")

dHOJE = DATE()

dDTFINAL = CTOD("06/08/2005")

- **LÓGICO:** Aceita apenas Verdadeiro (.T.) ou Falso (.F.)

Exemplo:

lCOMPLETO = .T.

lERRO = .F.

- **MEMO:** Variável do tipo Memo não existe, seria uma variável do tipo caracter com mais de 256 bytes que só caberia em um campo Memo.

Exemplo:

mOBS = MEMOREAD("NOTAS.TXT")

- **ARRAY:** Também chamado de matriz ou vetor. São como vários campos agrupados em uma lista. É uma estrutura de dados que contém uma série de dados ordenados, chamados "elementos". Os elementos são referenciados por número ordinal, primeiro elemento é 1, segundo 2... Os elementos podem ser de qualquer tipo, caracter, numérico, data etc. Veja mais detalhes em TUDO SOBRE ARRAYS.

- **CODEBLOCK:** É um tipo especial de variável que armazena um pedaço de código compilado.

Exemplo:

```
@ 10,10 SAY "CODIGO:" GET WCOD PICT "999999" VALID;
```

```
EVAL( { || WCOD := STRZERO(WCOD,6), .T.} )
```

Exemplos de Comandos Básicos e Laços de Repetição

@. . . 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:

@ 15,10 SAY ‘FATEC’

xcod:=0

@ 18,10 GET xcod

read

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

SALÁRIO := 3020.29

? STR(SALARIO,4) // resultado: 3020

? STR(SALARIO,8,3) // resultado: 3020.290

VAL( ) Propósito: Converter uma expressão caractere em um valor numérico. **Sintaxe:** VAL(<string>). **Exemplo:**

SALÁRIO := "2929.20"

? VAL (SALÁRIO) // resultado: 2929.20

TESTE := "COMPUTADOR"

? VAL(TESTE) // resultado: 0

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: 1) FOR I := 1 TO 100

@ 15,10 SAY 'CONTADOR:' +STR(I,3)

NEXT

2) FOR J := 100 TO 500 STEP 10

@ 18,05 SAY 'O VALOR DE J É '+ STRZERO(J,3)

NEXT

DO WHILE...ENDDO Propósito: Executa uma estrutura de controle enquanto uma condição for verdadeira. **Sintaxe:**

DO WHILE <condição>

<instruções>

[EXIT]

[LOOP]

ENDDO

Exemplo: 1)

```
DO WHILE .T.

xnumero:=0

@ 11,10 say ‘Digite um número’

@ 11,20 get xnumero

read

if empty(xnumero)

exit

endif

@ 13,10 say ‘o número digitado foi’+strzero(xnumero,3)

ENDDO
```

2) xresp:=’S’

```
DO WHILE XRESP#’N’ // # ou < > símbolos de diferente

xnome:=space(40)

@ 11,10 say ‘Nome: ‘

@ 11,25 get xnome

read

if lastkey()==27 // == exatamente igual

exit

endi

@ 15,18 say ‘O nome digitado foi: ‘+xnome

xresp:=space(01)

@ 20,10 say ‘Deseja continuar ?
```

@ 20,30 get xresp picture '!'

read

ENDDO

Exemplos de código

```
? "Alô Mundo!"
```

```
USE cliente SHARED NEW

CLEAR SCREEN

DO WHILE LASTKEY() != 27

   @ 01, 0 SAY "Codigo  " GET cliente->codigo PICT "999999" VALID cliente->codigo > 0
   @ 03, 0 SAY "Nome    " GET cliente->nome VALID ! empty(cliente->nome)
   @ 04, 0 SAY "Endereco" GET cliente->endereco

   READ

ENDDO
```

Obs: O código acima viola a regra básica do uso de tabelas em modo compartilhado.

```
@ . . . 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)
```

```
@. . . 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 // faça 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
```

```
AVERAGE
                Propósito:      Calcular a média aritmética de  campos  ou expressões de arquivos de dad
                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.
```

Ver também

- xBase
- dBase
- Visual FoxPro
- Harbour

Ligações externas

- Fórum Clipper On Line Compartilhamento de informações sobre xBase (http://www.pctoledo.com.br/forum) - Gratuito
- FlagShip Compilador compatível de Clipper para Linux, Unix e Windows (http://www.fship.com) - Comercial
- Alaska xBase++ Compilador compatível de Clipper para Windows (http://www.alaska-software.com) - Comercial
- Projeto Harbour Compilador compatível de Clipper para Linux, Unix e Windows (http://www.harbour-project.org) - Open Source
- xHarbour Compilador compatível de Clipper para Linux, Unix e Windows (http://www.xharbour.org/) - Open Source ou Comercial
- [1] (http://www.harbour-project.org/) - Projeto Harbour
- [2] (http://www.forumweb.com.br/foruns/index.php?/topic/12737-codigos-de-erro-do-clipper-internal-errors/) - Códigos de Erro do Clipper
- Clip Compilador de uma empresa russa compatível de Clipper para Linux (http://www.itk.ru/english/clip/clipchangelog.shtml) - Open Source

Obtida de "http://pt.wikipedia.org/wiki/Clipper_(linguagem_de_programa%C3%A7%C3%A3o)"
Categoria: Linguagens de programação