

Todos os direitos autorais reservados pela TOTVS S.A.

Proibida a reprodução total ou parcial, bem como a armazenagem em sistema de recuperação e a transmissão, de qualquer modo ou por qualquer outro meio, seja este eletrônico, mecânico, de fotocópia, de gravação, ou outros, sem prévia autorização por escrito da proprietária.

O desrespeito a essa proibição configura em apropriação indevida dos direitos autorais e patrimoniais da TOTVS.

Conforme artigos 122 e 130 da LEI no. 5.988 de 14 de Dezembro de 1973.

0073 – Programação ADVPL Avançado

Protheus
11





SUMÁRIO

OBJETIVOS DO CURSO	4
1. Programas de Atualização	5
1.1. Modelo1() ou AxCadastro()	7
1.2. Mbrowse()	9
1.3. MarkBrowse()	23
1.4. Modelo2()	29
1.5. Modelo3()	52
2. Relatórios não gráficos	73
2.1. Funções Utilizadas para Desenvolvimento de Relatórios	73
2.2. Estrutura de Relatórios Baseados na SetPrint()	80
2.3. Introdução à relatórios Gráficos	95
3. Manipulação de arquivos I	103
3.1. Geração e leitura de arquivos em formato texto	103
4. Oficina de programação I	122
4.1. Interfaces com sintaxe clássica	122
4.2. Réguas de processamento	125
4.3. ListBox().....	140
4.4. ScrollBox().....	147
4.5. ParamBox().....	151
5. Componentes da interface visual do ADVPL	159
5.1. Particularidades dos componentes visuais	166
6. Aplicações com a interface visual do ADVPL	169
6.1. Captura de informações simples (Multi-Gets)	169
6.2. Captura de múltiplas informações (Multi-Lines)	175
6.3. Barras de botões	197

7. Arredondamento	204
8. Utilização de Identação	205
9. Capitulação de Palavras-Chave	207
9.1. Palavras em maiúsculo	207
10. Utilização da Notação Húngara	208
11. Técnicas de programação eficiente	209

OBJETIVOS DO CURSO**Objetivos específicos do curso:**

Ao final do curso o treinando deverá ter desenvolvido os seguintes conceitos, habilidades e atitudes:

a) Conceitos a serem aprendidos

- Estruturas para implementação de relatórios e programas de atualização
- Estruturas para implementação de interfaces visuais
- Princípios do desenvolvimento de aplicações orientadas a objetos

b) Habilidades e técnicas a serem aprendidas

- Desenvolvimento de aplicações voltadas ao ERP Protheus
- Análise de fontes de média complexidade
- Desenvolvimento de aplicações básicas com orientação a objetos

c) Atitudes a serem desenvolvidas

- Adquirir conhecimentos através da análise dos funcionalidades disponíveis no ERP Protheus;
- Estudar a implementação de fontes com estruturas orientadas a objetos em ADVPL;
- Embasar a realização de outros cursos relativos a linguagem ADVPL

I. PROGRAMAS DE ATUALIZAÇÃO

Os programas de atualização de cadastros e digitação de movimentos seguem um padrão que se apóia no Dicionário de Dados.

Basicamente são três os modelos mais utilizados:

- **Modelo 1 ou AxCadastro:**

Para cadastramentos em tela cheia. Exemplo: Cadastro de Cliente.

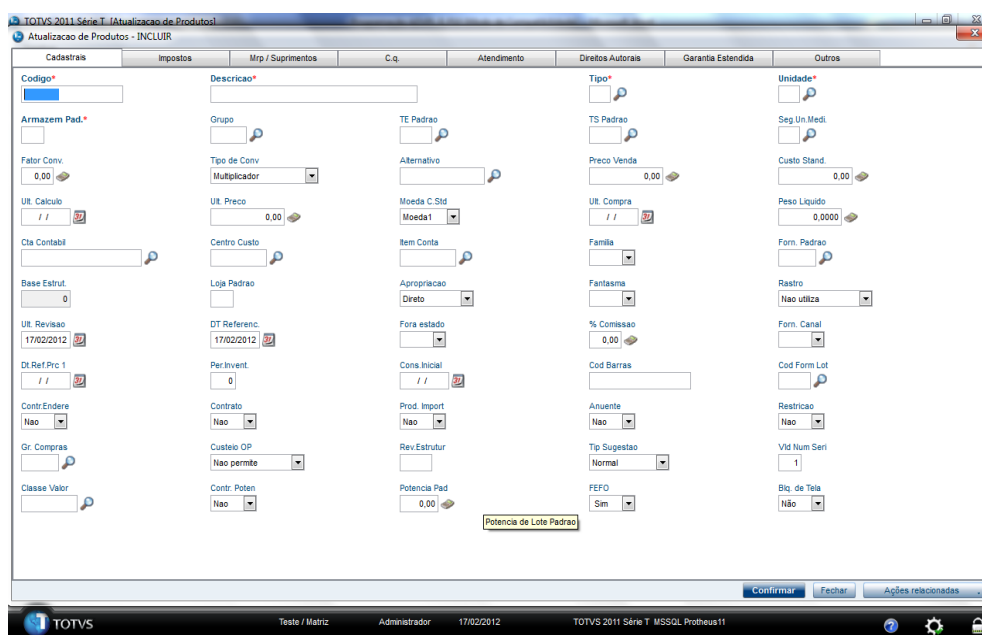
- **Modelo 2:**

Cadastramentos envolvendo apenas uma tabela, mas com um cabeçalho e, opcionalmente, um rodapé e um corpo com quantidade ilimitada de linhas. Ideal para casos em que há dados que se repetem por vários itens e que, por isso, são colocados no cabeçalho. Exemplo: Pedido de Compra.

- **Modelo 3:**

Cadastramentos envolvendo duas tabelas, um com dados de cabeçalho e outro digitado em linhas com os itens. Exemplo: Pedido de Vendas, Orçamento etc.

Todos os modelos são genéricos, ou seja, o programa independe da tabela a ser tratada, bastando praticamente que se informe apenas o seu Alias. O resto é obtido do Dicionário de Dados (SX3).



TOTVS 2011 Série T - (Pedido de Compra)
Pedido de Compra - ALTERAR

Numero: 000001 Data de Emissao: 26/01/2012 Fornecedor: 000001 Loja: 01
 Cond. Pagto: 001 A VISTA Contato: Filial p/ Entrega: 01
 Moeda: 1 REAL Taxa: 0,0000

Item	Produto	Unidade	Segunda UM	Tab. Preço	Quantidade	Preço Unitário	Vlr. Total	Qtd. 2a UM	Aliq. IPI	Numero da SC	Item da SC	Dt. Entrega	Armazen
0001	000001	UN	CX		10,00	10,00	100,00	0,83	15,00			26/01/2012	01

Totais Inf. Fornecedor Frete/Despesas Descontos Impostos Mensagem/Reajuste

Valor da Mercadoria: 100,00 Descontos: 0,00
 Frete: 0,00 Despesas: 0,00
 Seguro: 0,00
Total do Pedido: 115,00

Confirmar Fechar Ações relacionadas

TOTVS Teste / Matriz Administrador 17/02/2012 TOTVS 2011 Série T MSSQL Protheus11 F4 | F5

TOTVS 2011 Série T - (Pedido de Compra)
Pedido de Compra - ALTERAR

Numero: 000001 Data de Emissao: 26/01/2012 Fornecedor: 000001 Loja: 01
 Cond. Pagto: 001 A VISTA Contato: Filial p/ Entrega: 01
 Moeda: 1 REAL Taxa: 1,0000

Item	Produto	Unidade	Segunda UM	Tab. Preço	Quantidade	Preço Unitário	Vlr. Total	Qtd. 2a UM	Aliq. IPI	Numero da SC	Item da SC	Dt. Entrega	Armazen
0001	000001	UN	CX		10,00	10,00	100,00	0,83	15,00			26/01/2012	01
0002	000002	UN			20,00	10,00	200,00	0,00	0,00			17/02/2012	01

Totais Inf. Fornecedor Frete/Despesas Descontos Impostos Mensagem/Reajuste

Valor da Mercadoria: 300,00 Descontos: 0,00
 Frete: 0,00 Despesas: 0,00
 Seguro: 0,00
Total do Pedido: 315,00

Confirmar Fechar Ações relacionadas

TOTVS Teste / Matriz Administrador 17/02/2012 TOTVS 2011 Série T MSSQL Protheus11 F4 | F5

1.1. Modelo1() ou AxCadastro()

O AxCadastro() é uma funcionalidade de cadastro simples, com poucas opções de customização, a qual é composta de:

- Browse padrão para visualização das informações da base de dados, de acordo com as configurações do SX3 – Dicionário de Dados (campo browse).
- Funções de pesquisa, visualização, inclusão, alteração e exclusão padrões para visualização de registros simples, sem a opção de cabeçalho e itens.
 - Sintaxe: AxCadastro(cAlias, cTitulo, cVldExc, cVldAlt)
 - Parâmetros:

cAlias	Alias padrão do sistema para utilização, o qual deve estar definido no dicionário de dados – SX3.
cTitulo	Título da Janela
cVldExc	Validação para Exclusão
cVldAlt	Validação para Alteração

Exemplo: Função AxCadastro()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função | XCADSA2 | Autor | ARNALDO RAYMUNDO JR. | Data | |
```

```
+-----
```

```
| Descrição | Exemplo de utilização da função AXCADASTRO()
```

```
|
```

```
|+-----
```

```
| Uso | Curso ADVPL |
```

```
|+-----
```

```
/*/
```

```
User Function XCadSA2()
```

```
Local cAlias := "SA2"
```

```
Local cTitulo := "Cadastro de Fornecedores"
```

```
Local cVldExc := ".T."
```

Local cVldAlt := ".T."

dbSelectArea(cAlias)

dbSetOrder(1)

AxCadastro(cAlias,cTitulo,cVldExc,cVldAlt)

Return Nil

Exercício

Desenvolver um AxCadastro para a tabela padrão do ERP – SB1: Produtos

Exemplo: Função de validação da alteração

/*/

+-----

Função	VLDALT	Autor	ARNALDO RAYMUNDO JR.	Data	
--------	--------	-------	----------------------	------	--

+-----

Descrição	Função de validação de alteração para a AXCADASTRO()
-----------	---

+-----

Uso	Curso ADVPL
-----	-------------

+-----

/*/

User Function VldAlt(cAlias,nReg,nOpc)

Local lRet := .T.

Local aArea := GetArea()

Local nOpcao := 0

nOpcao := AxAlterar(cAlias,nReg,nOpc)

If nOpcao == 1

MsgInfo("Ateração concluída com sucesso!")

Endif

RestArea(aArea)

Return lRet

Exemplo: Função de validação da exclusão

```

/*/
+-----+
| Função | VLDEXC   | Autor | ARNALDO RAYMUNDO JR. | Data |      |
+-----+
| Descrição                                     | Função de validação de exclusão para a
AXCADASTRO() |
+-----+
| Uso                                           | Curso ADVPL
|
+-----+
/*/

```

User Function VldExc(cAlias,nReg,nOpc)

```

Local lRet           := .T.
Local aArea          := GetArea()
Local nOpcao         := 0

```

nOpcao := AxExclui(cAlias,nReg,nOpc)

```

If nOpcao == 1
    MsgInfo("Exclusão concluída com sucesso!")
Endif

```

```

RestArea(aArea)
Return lRet

```

Exercício

Implementar no AxCadastro as validações de alteração e exclusão.

1.2. Mbrowse()

A Mbrowse() é uma funcionalidade de cadastro que permite a utilização de recursos mais aprimorados na visualização e manipulação das informações do sistema, possuindo os seguintes componentes:

- Browse padrão para visualização das informações da base de dados, de acordo com as configurações do SX3 – Dicionário de Dados (campo browse).
- Parametrização para funções específicas para as ações de visualização, inclusão, alteração e exclusão de informações, o que viabiliza a manutenção de informações com estrutura de cabeçalhos e itens.
- Recursos adicionais como identificadores de status de registros, legendas e filtros para as informações.
 - Sintaxe: MBrowse(nLin1, nCol1, nLin2, nCol2, cAlias, aFixe, cCpo, nPar08, cFun, nClickDef, aColors, cTopFun, cBotFun, nPar14, blnitBloc, lNoMnuFilter, lSeeAll, lChgAll)
 - Parâmetros:

nLin1	Número da Linha Inicial
nCol1	Número da Coluna Inicial
nLin2	Número da Linha Final
nCol2	Número da Coluna Final
cAlias	Alias do arquivo que será visualizado no browse. Para utilizar a função MBrowse com arquivos de trabalho, o alias do arquivo de trabalho deve ser obrigatoriamente 'TRB' e o parâmetro aFixe torna-se obrigatório.
aFixe	<p>Array bi-dimensional contendo os nomes dos campos fixos pré-definidos, obrigando a exibição de uma ou mais colunas ou a definição das colunas quando a função é utilizada com arquivos de trabalho.</p> <p>A estrutura do array é diferente para arquivos que fazem parte do dicionário de dados e para arquivos de trabalho.</p> <p>Arquivos que fazem parte do dicionários de dados</p> <p>[n][1]=>Descrição do campo [n][2]=>Nome do campo</p>
aFixe	<p>Arquivos de trabalho</p> <p>[n][1]=>Descrição do campo [n][2]=>Nome do campo [n][3]=>Tipo [n][4]=>Tamanho [n][5]=>Decimal [n][6]=>Picture</p>

- Parâmetros:

cCpo	Campo a ser validado se está vazio ou não para exibição do bitmap de status. Quando esse parâmetro é utilizado, a primeira coluna do browse será um bitmap indicando o status do registro, conforme as condições configuradas nos parâmetros cCpo , cFun e aColors .
nPar08	Parâmetro reservado.
cFun	Função que retornará um valor lógico para exibição do bitmap de status. Quando esse parâmetro é utilizado, o parâmetro cCpo é automaticamente desconsiderado.
nClickDef	Número da opção do aRotina que será executada quando for efetuado um duplo clique em um registro do browse. O default é executar a rotina de visualização.
aColors	Array bi-dimensional para possibilitar o uso de diferentes bitmaps de status. [n][1]=>Função que retornará um valor lógico para a exibição do bitmap [n][2]=>Nome do bitmap que será exibido quando a função retornar .T. (True). O nome do bitmap deve ser um resource do repositório e quando esse parâmetro é utilizado os parâmetros cCpo e cFun são automaticamente desconsiderados.
cTopFun	Função que retorna o limite superior do filtro baseado na chave de índice selecionada. Esse parâmetro deve ser utilizado em conjunto com o parâmetro cBotFun .
cBotFun	Função que retorna o limite inferior do filtro baseado na chave de índice selecionada. Esse parâmetro deve ser utilizado em conjunto com o parâmetro cTopFun .
nPar14	Parâmetro reservado.
bInitBloc	Bloco de código que será executado no ON INIT da janela do browse. O bloco de código receberá como parâmetro o objeto da janela do browse.
INoMnuFilter	Valor lógico que define se a opção de filtro será exibida no menu da MBrowse. .T. => Não exibe a opção no menu .F. => (default) Exibe a opção no menu. A opção de filtro na MBrowse está disponível apenas para TopConnect.

ISeeAll	Identifica se o Browse deverá mostrar todas as filiais. O valor default é .F. (False), não mostra todas as filiais. Caso os parâmetros cTopFun ou cBotFun sejam informados esse parâmetro será configurado automaticamente para .F. (False) Parâmetro válido à partir da versão 8.11 . A função SetBrwSeeAll muda o valor default desse parâmetro.
IchgAll	Identifica se o registro de outra filial está autorizado para alterações. O valor default é .F. (False), não permite alterar registros de outras filiais. Quando esse parâmetro está configurado para .T. (True), o parâmetro ISeeAll é configurado automaticamente para .T. (True). Caso os parâmetros cTopFun ou cBotFun sejam informados esse parâmetro será configurado automaticamente para .F. (False). Parâmetro válido à partir da versão 8.11 . A função SetBrwChgAll muda o valor default desse parâmetro.

- Variáveis private adicionais

aRotina	Array contendo as funções que serão executadas pela Mbrowse, nele será definido o tipo de operação a ser executada (inclusão, alteração, exclusão, visualização, pesquisa, etc.), e sua estrutura é composta de 5 (cinco) dimensões: [n][1] - Título; [n][2] – Rotina; [n][3] – Reservado; [n][4] – Operação (1 - pesquisa; 2 - visualização; 3 - inclusão; 4 - alteração; 5 - exclusão); Ele ainda pode ser parametrizado com as funções básicas da AxCadastro conforme abaixo: AADD(aRotina,{"Pesquisar" ,"AxPesqui",0,1}) AADD(aRotina,{"Visualizar" ,"AxVisual",0,2}) AADD(aRotina,{"Incluir" ,"AxInclui",0,3}) AADD(aRotina,{"Alterar" ,"AxAltera",0,4}) AADD(aRotina,{"Excluir" ,"AxDeleta",0,5})
cCadastro	Título do browse que será exibido.

- Informações passadas para funções do aRotina:

Ao definir as funções no array aRotina, se o nome da função não for especificado com “()”, a Mbrowse passará como parâmetros as seguintes variáveis de controle:

cAlias	Nome da área de trabalho definida para a Mbrowse
---------------	--

nReg	Recno do registro posicionado no Browse
nOpc	Posição da opção utilizada na Mbrowse de acordo com a ordem da função no array aRotina.

Importante

A posição das funções no array aRotina define o conteúdo de uma variável de controle que será repassada para as funções chamadas a partir da Mbrowse, convencionada como nOpc. Desta forma, para manter o padrão da aplicação ERP a ordem a ser seguida na definição do aRotina é:

1. Pesquisar
2. Visualizar
3. Incluir
4. Alterar
5. Excluir
6. Livre

Exemplo: Função Mbrowse()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função | MBRWSA1 | Autor | ARNALDO RAYMUNDO JR. | Data | |
```

```
+-----
```

```
| Descrição | Exemplo de utilização da função MBROWSE() |
```

```
+-----
```

```
| Uso | Curso ADVPL
```

```
+-----
```

```
+-----
```

```
/*/
```

```
User Function MBrwSA1()
```

```
Local cAlias := "SA1"
```

```
Private cCadastro := "Cadastro de Clientes"
```

```
Private aRotina := {}
```

```

AADD(aRotina,{"Pesquisar"           ,"AxPesqui",0,1})
AADD(aRotina,{"Visualizar"          ,"AxVisual",0,2})
AADD(aRotina,{"Incluir"              ,"AxInclui",0,3})
AADD(aRotina,{"Alterar"              ,"AxAlterar",0,4})
AADD(aRotina,{"Excluir"              ,"AxDeleta",0,5})

```

```

dbSelectArea(cAlias)
dbSetOrder(1)
mBrowse(6,1,22,75,cAlias)

```

```
Return Nil
```

Exemplo: Função Inclui() substituindo a função AxInclui() – Chamada da Mbrowse()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

Função	MBRWSA1	Autor	ARNALDO RAYMUNDO JR.	Data	
--------	---------	-------	----------------------	------	--

```
+-----
```

Descrição	Exemplo de utilização da função MBROWSE()

```
+-----
```

Uso	Curso ADVPL

```
+-----
```

```
/*/
```

```
User Function MBrwSA1()
```

```
Local cAlias := "SA1"
```

```
Private cCadastro := "Cadastro de Clientes"
```

```
Private aRotina := {}
```

```

AADD(aRotina,{"Pesquisar"           ,"AxPesqui"   ,0,1})
AADD(aRotina,{"Visualizar"          ,"AxVisual"   ,0,2})

```

```
AADD(aRotina,{"Incluir" , "U_Inclui" ,0,3})
```

```
AADD(aRotina,{"Alterar" , "AxAlterar" ,0,4})
```

```
AADD(aRotina,{"Excluir" , "AxDeleta" ,0,5})
```

```
dbSelectArea(cAlias)
```

```
dbSetOrder(1)
```

```
mBrowse(6,1,22,75,cAlias)
```

```
Return Nil
```

Exemplo: Função Inclui() substituindo a função AxInclui() – Função Inclui()

```
/*/
```

```
+-----
```

Função	INCLUI	Autor	ARNALDO RAYMUNDO JR.	Data	
--------	--------	-------	----------------------	------	--

```
+-----
```

Descrição	Função de inclusão específica chamando a AXINCLUI()
-----------	---

```
+-----
```

Uso	Curso ADVPL
-----	-------------

```
+-----
```

```
/*/
```

```
User Function Inclui(cAlias, nReg, nOpc)
```

```
Local cTudoOk := "(Alert('OK'),.T.)"
```

```
Local nOpcaco := 0
```

```
nOpcaco := AxInclui(cAlias,nReg,nOpc,,,,cTudoOk)
```

```
If nOpcaco == 1
```

```
    MsgInfo("Inclusão concluída com sucesso!")
```

```
Elseif == 2
```

```
    MsgInfo("Inclusão cancelada!")
```

```
Endif
```

```
Return Nil
```

Exemplo: Determinando a opção do aRotina pela informação recebida em nOpc

```
#include "protheus.ch"
/*/
+-----+
| Função | EXCLUI      | Autor | ARNALDO RAYMUNDO JR. | Data |      |
+-----+
| Descrição                                     | Função de exclusão específica chamando a
AxDeleta      |
+-----+
| Uso      | Curso ADVPL                                     |
+-----+
/*/

User Function Exclui(cAlias, nReg, nOpc)

Local cTudoOk := "(Alert('OK'),.T.)"
Local nOpcao := 0

nOpcao := AxDeleta(cAlias,nReg,aRotina[nOpc,4])
// Identifica corretamente a opção definida para o função em aRotinas com mais // do que os
5 elementos padrões.






If nOpcao == 1
    MsgInfo("Exclusão realizada com sucesso!")
Elseif      == 2
    MsgInfo("Exclusão cancelada!")
Endif

Return Nil
```

1.2.1. AxFunctions()

Conforme mencionado nos tópicos sobre as interfaces padrões AxCadastro() e Mbrowse(), existem funções padrões da aplicação ERP que permitem a visualização, inclusão, alteração e exclusão de dados em formato simples.

Estas funções são padrões na definição da interface AxCadastro() e podem ser utilizadas também da construção no array aRotina utilizado pela Mbrowse(), as quais estão listadas a seguir:

-  **AXPESQUI()**
-  **AXVISUAL()**
-  **AXINCLUI()**
-  **AXALTERA()**
-  **AXDELETA()**

AXPESQUI()

Sintaxe	AXPESQUI()
Descrição	Função de pesquisa padrão em registros exibidos pelos browses do sistema, a qual posiciona o browse no registro pesquisado. Exibe uma tela que permite a seleção do índice a ser utilizado na pesquisa e a digitação das informações que compõe a chave de busca.

AXVISUAL()

Sintaxe	AXVISUAL(cAlias, nReg, nOpc, aAcho, nColMens, cMensagem, cFunc,; aButtons, lMaximized)
Descrição	Função de visualização padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

AXINCLUI()

Sintaxe	AxInclui(cAlias, nReg, nOpc, aAcho, cFunc, aCpos, cTudoOk, lF3,; cTransact, aButtons, aParam, aAuto, lVirtual, lMaximized)
Descrição	Função de inclusão padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

AXALTERA()

Sintaxe	AxAlterar(cAlias, nReg, nOpc, aAcho, cFunc, aCpos, cTudoOk, lF3,; cTransact, aButtons, aParam, aAuto, lVirtual, lMaximized)
Descrição	Função de alteração padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

AXDELETA()

Sintaxe	AXDELETA(cAlias, nReg, nOpc, cTransact, aCpos, aButtons, aParam,; aAuto, lMaximized)
Descrição	Função de exclusão padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

Exercício

Implementar uma MBrowse com as funções de cadastro padrões para a tabela padrão do ERP – SB1: Produtos

1.2.2. FilBrowse()

A FilBrowse() é uma funcionalidade que permite a utilização de filtros na MBrowse().

- Sintaxe: FilBrowse(cAlias, aQuery, cFiltro, lShowProc)
- Parâmetros:

cAlias	Alias ativo definido para a Mbrowse()
aQuery	Este parâmetro deverá ser inicializado sempre vazio e sua passagem obrigatoriamente por referência, pois, seu retorno será enviado para a função EndFilBrw(). [1]=>Nome do Arquivo Físico [2]=>Ordem correspondente ao Sindex
cFiltro	Condição de filtro para a MBrowse()
lShowProc	Habilita (.T.) ou desabilita (.F.) a apresentação da mensagem "Selecionando registros ...", no processamento.

1.2.3. EndFilBrw()

A EndFilBrw() é uma funcionalidade que permite eliminar o filtro e o arquivo temporário criados pela FilBrowse().

- Sintaxe: EndFilBrw(cAlias, aQuery)
- Parâmetros:

cAlias	Alias ativo definido para a Mbrowse()
aQuery	Array de retorno passado por referência para a FilBrowse(). [1]=>Nome do Arquivo Físico [2]=>Ordem correspondente ao Sindex

1.2.4. PesqBrw()

A PesqBrw() é uma funcionalidade que permite a pesquisa dentro da MBrowse(). Esta função deverá obrigatoriamente substituir a função AxPesqui, no array do aRotina, sempre que for utilizada a função FilBrowse().

- Sintaxe: PesqBrw(cAlias , nReg, bBrwFilter)

- Parâmetros:

cAlias	Alias ativo definido para a Mbrowse()
nReg	Número do registro
bBrwFilter	Bloco de Código que contém a FilBrowse() Ex: bBrwFilter := { FilBrowse(cAlias, aQuery, cFiltro, IShowProc) }

1.2.5. BrwLegenda ()

A BrwLegenda() é uma funcionalidade que permite a inclusão de legendas na MBrowse().

- Sintaxe: BrwLegenda(cCadastro , cTitulo, aLegenda)
- Parâmetros:

cCadastro	Mesma variável utilizada para a MBrowse, que identifica o cadastro que está em uso no momento
cTitulo	Título (identificação) da Legenda
aLegenda	Array contendo de definição da cor e do texto, explicativo sobre o que ela representa na MBrowse Ex: {{ "Cor", "Texto" }}

Importante

Lista de cores disponíveis no Protheus

- | | |
|--------------|---------------|
| • BR_AMARELO | • BR_MARRON |
| • BR_AZUL | • BR_VERDE |
| • BR_BRANCO | • BR_VERMELHO |
| • BR_CINZA | • BR_PINK |
| • BR_LARANJA | • BR_PRETO |

Exemplo: Mbrowse() utilizando as funções acessórias

```
#Include "Protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Programa                                | MBrwSA2    | Autor | SERGIO FUZINAKA | Data
```

```
|                                     |            |
```

```
+-----
```

Descrição	Exemplo da MBrowse utilizando a tabela de
Cadastro de	
	Fornecedores
Uso	Curso de ADVPL

/*/

User Function MBrwSA2()

Local cAlias := "SA2"

Local aCores := {}

Local cFiltro := "A2_FILIAL == '"+xFilial('SA2')+"' .And. A2_EST == 'SP'"

Private cCadastro := "Cadastro de Fornecedores"

Private aRotina := {}

Private aIndexSA2 := {}

Private bFiltroBrw:= { | FilBrowse(cAlias,@aIndexSA2,@cFiltro) }

AADD(aRotina,{"Pesquisar" , "PesqBrw" ,0,1})

AADD(aRotina,{"Visualizar" , "AxVisual" ,0,2})

AADD(aRotina,{"Incluir" , "U_BInclui" ,0,3})

AADD(aRotina,{"Alterar" , "U_BAltera" ,0,4})

AADD(aRotina,{"Excluir" , "U_BDeleta" ,0,5})

AADD(aRotina,{"Legenda" , "U_BLegenda" ,0,3})

/*

-- CORES DISPONIVEIS PARA LEGENDA --

BR_AMARELO

BR_AZUL

BR_BRANCO

BR_CINZA

BR_LARANJA

BR_MARRON

BR_VERDE

```
BR_VERMELHO
```

```
BR_PINK
```

```
BR_PRETO
```

```
*/
```

```
AADD(aCores,{"A2_TIPO == 'F'"           ,"BR_VERDE"           })
```

```
AADD(aCores,{"A2_TIPO == 'J'"           ,"BR_AMARELO"          })
```

```
AADD(aCores,{"A2_TIPO == 'X'"           ,"BR_LARANJA" })
```

```
AADD(aCores,{"A2_TIPO == 'R'"           ,"BR_MARRON"           })
```

```
AADD(aCores,{"Empty(A2_TIPO)"           ,"BR_PRETO"            })
```

```
dbSelectArea(cAlias)
```

```
dbSetOrder(1)
```

```
//+-----
```

```
//| Cria o filtro na MBrowse utilizando a função FilBrowse
```

```
//+-----
```

```
Eval(bFiltrarBrw)
```

```
dbSelectArea(cAlias)
```

```
dbGoTop()
```

```
mBrowse(6,1,22,75,cAlias,,,,,aCores)
```

```
//+-----
```

```
//| Deleta o filtro utilizado na função FilBrowse
```

```
//+-----
```

```
EndFilBrw(cAlias,aIndexSA2)
```

```
Return Nil
```

```
//+-----
```

```
//| Função: BInclui - Rotina de Inclusão
```

```
//+-----
```

```
User Function BInclui(cAlias,nReg,nOpc)
```

```
Local nOpcao := 0
```

```
nOpcao := AxInclui(cAlias,nReg,nOpc)
```

```
If nOpcao == 1
```

```
    MsgInfo("Inclusão efetuada com sucesso!")
```

```
Else
```

```
    MsgInfo("Inclusão cancelada!")
```

```
Endif
```

```
Return Nil
```

```
//+-----
```

```
//|Função: BAltera - Rotina de Alteração
```

```
//+-----
```

```
User Function BAltera(cAlias,nReg,nOpc)
```

```
Local nOpcao := 0
```

```
nOpcao := AxAltera(cAlias,nReg,nOpc)
```

```
If nOpcao == 1
```

```
    MsgInfo("Alteração efetuada com sucesso!")
```

```
Else
```

```
    MsgInfo("Alteração cancelada!")
```

```
Endif
```

```
Return Nil
```

```
//+-----
```

```
//|Função: BDeleta - Rotina de Exclusão
```

```
//+-----
```

```
User Function BDeleta(cAlias,nReg,nOpc)
```

```
Local nOpcao := 0
```

```
nOpcao := AxDeleta(cAlias,nReg,nOpc)
```

```
If nOpcao == 1
```

```
        MsgInfo("Exclusão efetuada com sucesso!")
    Else
        MsgInfo("Exclusão cancelada!")
    Endif

Return Nil

//+-----
//| Função: BLegenda - Rotina de Legenda
//+-----
User Function BLegenda()

Local aLegenda := {}

AADD(aLegenda,{"BR_VERDE"           ,"Pessoa Física" })
AADD(aLegenda,{"BR_AMARELO"         ,"Pessoa Jurídica" })
AADD(aLegenda,{"BR_LARANJA"         ,"Exportação"      })
AADD(aLegenda,{"BR_MARRON"          ,"Fornecedor Rural" })
AADD(aLegenda,{"BR_PRETO"           ,"Não Classificado" })

BrwLegenda(cCadastro, "Legenda", aLegenda)

Return Nil
```

Exercício

Implementar a legenda para a MBrowse da tabela padrão do ERP – SB1: Produtos

1.3. MarkBrowse()

A função MarkBrow() permite que os elementos de um browse, sejam marcados ou desmarcados. Para utilização da MarkBrow() é necessário declarar as variáveis cCadastro e aRotina como Private, antes da chamada da função.

- Sintaxe: MarkBrow (cAlias, cCampo, cCpo, aCampos, lInvert, cMarca, cCtrlM, uPar8, cExpIni, cExpFim, cAval, bParBloco)
- Parâmetros:

cAlias	Alias ativo definido para a Mbrowse()
cCampo	Campo do arquivo onde será feito o controle (gravação) da marca.
cCpo	Campo onde será feita a validação para marcação e exibição do bitmap de status.
aCampos	Vetor de colunas a serem exibidas no browse, deve conter as seguintes dimensões: [n][1] – nome do campo; [n][2] - Nulo (Nil); [n][3] - Título do campo; [n][4] - Máscara (picture).
lInvert	Inverte a marcação.
cMarca	String a ser gravada no campo especificado para marcação.
cCtrlM	Função a ser executada caso deseje marcar todos elementos.
uPar8	Parâmetro reservado.
cExpIni	Função que retorna o conteúdo inicial do filtro baseada na chave de índice selecionada.
cExpFim	Função que retorna o conteúdo final do filtro baseada na chave de índice selecionada.
cAval	Função a ser executada no duplo clique em um elemento no browse.
bParBloco	Bloco de código a ser executado na inicialização da janela

- Informações passadas para funções do aRotina:

Ao definir as funções no array aRotina, se o nome da função não for especificado com “()”, a MarkBrowse passará como parâmetros as seguintes variáveis de controle:

cAlias	Nome da área de trabalho definida para a Mbrowse
nReg	Recno do registro posicionado no Browse
nOpc	Posição da opção utilizada na Mbrowse de acordo com a ordem da função no array a Rotina.
cMarca	Marca em uso pela MarkBrw()
lInverte	Indica se foi utilizada a inversão da seleção dos itens no browse.

1.3.1. Funções de Apoio

GetMark: define a marca atual.

IsMark: avalia se um determinado conteúdo é igual a marca atual.

ThisMark: captura a marca em uso.

ThisInv: indica se foi usado o recurso de selecionar todos (inversão).

MarkBRefresh: atualiza exibição na marca do browse. Utilizada quando a marca é colocada ou removida em blocos e não pelo clique.

Exemplo: Função MarkBrow() e acessórias

```
#include "protheus.ch"
/*
+-----+
| Programa                                | MkBrwSA1   | Autor | ARNALDO RAYMUNDO
JR. | Data |                               |
+-----+
| Desc.   | MarkBrowse Genérico                                |
+-----+
| Uso                                | Curso de ADVPL
|
+-----+
*/
```

USER FUNCTION MkBrwSA1()

```
Local aCpos                := {}
Local aCampos              := {}
Local nl                   := 0
Local cAlias               := "SA1"

Private aRotina            := {}
Private cCadastro          := "Cadastro de Clientes"
Private aRecSel            := {}

AADD(aRotina,{"Pesquisar"    ,"AxPesqui"    ,0,1})
AADD(aRotina,{"Visualizar"   ,"AxVisual"   ,0,2})
AADD(aRotina,{"Incluir"      ,"AxInclui"   ,0,3})
AADD(aRotina,{"Alterar"      ,"AxAltera"   ,0,4})
AADD(aRotina,{"Excluir"      ,"AxDeleta"   ,0,5})
AADD(aRotina,{"Visualizar Lote","U_VisLote"  ,0,5})
AADD(aCpos,"A1_OK"          )
AADD(aCpos,"A1_FILIAL"      )
```

```

AADD(aCpos, "A1_COD" )
AADD(aCpos, "A1_LOJA" )
AADD(aCpos, "A1_NOME" )
AADD(aCpos, "A1_TIPO" )

dbSelectArea("SX3")
dbSetOrder(2)
For nl := 1 To Len(aCpos)
    IF dbSeek(aCpos[nl])

        AADD(aCampos,{X3_CAMPO,"",IIF(nl==1,"",Trim(X3_TITULO)),,
                    Trim(X3_PICTURE))

    ENDIF
Next

DbSelectArea(cAlias)
DbSetOrder(1)

MarkBrow(cAlias,aCpos[1],"A1_TIPO == ' '",aCampos,.F.,GetMark(,"SA1","A1_OK"))

Return Nil

```

Exemplo: Função VisLote() – utilização das funções acessórias da MarkBrow()

```

/*/
+-----+
| Programa                | VisLote()   | Autor | ARNALDO RAYMUNDO
JR. | Data |                |
+-----+
| Desc.   | Função utilizada para demonstrar o uso do recurso da MarkBrowse |
+-----+
| Uso     | Curso de ADVPL
|
+-----+
/*/

USER FUNCTION VisLote()

```

```

Local cMarca                := ThisMark()
Local nX  := 0
Local lInvert                := ThisInv()
Local cTexto                 := ""
Local cEOL                   := CHR(10)+CHR(13)
Local oDlg
Local oMemo

DbSelectArea("SA1")
DbGoTop()
While SA1->(!EOF())

    // IsMark("A1_OK", cMarca, lInverte)
    IF SA1->A1_OK == cMarca .AND. !lInvert
        AADD(aRecSel,{SA1->(Recno()),SA1->A1_COD, SA1->A1_LOJA, SA1->A1_NREDUZ})
    ELSEIF SA1->A1_OK != cMarca .AND. lInvert
        AADD(aRecSel,{SA1->(Recno()),SA1->A1_COD,SA1->A1_LOJA, SA1->A1_NREDUZ})
    ENDIF
    SA1->(dbSkip())

Enddo

IF Len(aRecSel) > 0
    cTexto := "Código | Loja | Nome Reduzido      "+cEol
    //
    "1234567890123456789012345678901234567890
    //                                "CCCCC | LL | NNNNNNNNNNNNNNNNNNNNNNNN
+cEol

    For nX := 1 to Len(aRecSel)

        cTexto += aRecSel[nX][2]+Space(1)+ "| "+Space(2) + aRecSel[nX][3]+Space(3)+" | "
        cTexto += Space(1)+SUBSTRING(aRecSel[nX][4],1,20)+Space(1)
        cTexto += cEOL

    Next nX

DEFINE MSDIALOG oDlg TITLE "Clientes Selecionados" From 000,000 TO 350,400

```

PIXEL

```
@ 005,005 GET oMemo VAR cTexto MEMO SIZE 150,150 OF oDlg PIXEL
oMemo:BRClicked := {} | AllwaysTrue()
DEFINE SBUTTON FROM 005,165 TYPE 1 ACTION oDlg:End() ENABLE OF oDlg PIXEL
ACTIVATE MSDIALOG oDlg CENTER
  LimpaMarca()
```

ENDIF

RETURN

Exemplo: Função LimpaMarca() – utilização das funções acessórias da MarkBrow()

```
/*
+-----+
| Programa                               | LimpaMarca | Autor | ARNALDO RAYMUNDO JR.
| Data |                               |
+-----+
| Desc.   | Função utilizada para demonstrar o uso do recurso da MarkBrowse |
+-----+
| Uso     | Curso de ADVPL
|
+-----+

*/

STATIC FUNCTION LimpaMarca()
Local nX := 0
  For nX := 1 to Len(aRecSel)
    SA1->(DbGoto(aRecSel[nX][1]))
    RecLock("SA1",.F.)
    SA1->A1_OK := SPACE(2)
    MsUnLock()
  Next nX

RETURN
```

Exercício

Implementar uma MarkBrowse com as funções de cadastro padrões para a tabela padrão do ERP – SB1: Produtos

Exercício

Implementar na MarkBrowse da tabela padrão SB1, uma função para exclusão de múltiplos itens selecionados no Browse. ERP – SB1: Produtos

Exercício

Implementar uma MarkBrowse para a tabela SA1, para visualização de dados de múltiplos clientes selecionados

1.4. Modelo2()

O nome Modelo 2 foi conceituado pela Microsiga por se tratar de um protótipo de tela para entrada de dados. Inicialmente vamos desmistificar dois pontos:

- Função Modelo2() – Trata-se de uma função pronta que contempla o protótipo Modelo 2, porém, este é um assunto que não iremos tratar aqui, visto que é uma funcionalidade simples que quando necessário intervir em algo na rotina não há muito recurso para tal.
- Protótipo Modelo 2 – Trata-se de uma tela, onde seu objetivo é efetuar a manutenção em vários registros de uma só vez. Por exemplo: efetuar o movimento interno de vários produtos do estoque em um único lote.

1.4.1. Componentes de uma tela no formato Modelo 2

Objeto MsDialog()

Deve ser utilizada como janela padrão para entrada de dados, é um tipo de objeto modal, ou seja, não permite que outra janela ativa receba dados enquanto esta estiver ativa.

Toda vez que utilizar este comando o ADVPL exige que seja declarado a diretiva Include no cabeçalho do programa o arquivo "Protheus.ch", isto porque o compilador precisará porque este comando trata-se de um pseudo código e sua tradução será feita na compilação. Vale lembrar também que este só será acionado depois que instanciado e ativado por outro comando.

```
DEFINE MSDIALOG oDlg TITLE "Protótipo Modelo 2" FROM 0,0 TO 280,552 OF;  
oMainWnd PIXEL  
  
ACTIVATE MSDIALOG oDlg CENTER
```

Reparem que o comando DEFINE MSDIALOG instanciou e o comando ACTIVATE MSDIALOG ativa todos os objetos, ou seja, todo ou qualquer outro objeto que precisar colocar nesta janela será preciso informar em qual objeto, para este caso sempre será utilizada a variável de objeto exportável **oDlg**.

Função EnchoiceBar()

Função que cria uma barra de botões padrão de Ok e Cancelar, permitindo a implementação de botões adicionais.

- Sintaxe: ENCHOICEBAR(oDlg, bOk, bCancelar, [IMensApag] , [aBotoes])
- Parâmetros:

oDlg	Objeto	Janela onde a barra será criada.
bOk	Objeto	Bloco de código executado quando clicado botão Ok.
bCancelar	Objeto	Bloco de código executado quando clicado.
IMensApag	Lógico	Indica se ao clicar no botão Ok aparecerá uma tela de confirmação de exclusão. Valor padrão falso.
aBotões	Vetor	Vetor com informações para criação de botões adicionais na barra. Seu formato é {bitmap, bloco de código, mensagem}.

Figura: Protótipo Modelo2 – Enchoice

Objeto TPanel()

Repare que para facilitar o desenvolvimento foi utilizado o objeto TPanel para ajudar o alinhamento dos objetos TSay e TGet, ou seja, a utilização deste recurso permite que o programador não se preocupe com coordenadas complexas para deixar a união dos objetos simétricos.

Utilize o objeto TPanel quando desejar criar um painel estático, onde podem ser criados outros controles com o objetivo de organizar ou agrupar componentes visuais.

- Sintaxe: TPanel():New([anRow], [anCol], [acText], [aoWnd], [aoFont], [alCentered], [IPar6], [anClrText], [anClrBack], [anWidth], [anHeight], [alLowered], [alRaised])
- Parâmetros:

nRow	Numérico vertical em pixel.
nCol	Numérico horizontal em pixel.
cText	Texto a ser exibido ao fundo.
oWnd	Objeto da janela ou controle onde será criado o objeto.
oFont	Características da fonte do texto que aparecerá ao fundo.

ICentered	Exibe o texto do título centralizado.
IPar6	Reservado.
nClrText	Cor do texto de controle.
nClrBack	Cor do fundo de controle.
nWidth	Largura do controle em pixel.
nHeight	Altura do controle em pixel.
ILowered	Exibe o painel rebaixado em relação ao controle de fundo.
IRaised	Exibe a borda do controle rebaixado em relação ao controle de fundo.

Comando SAY - Objeto: TSay()

O comando SAY ou objeto TSay exibe o conteúdo de texto estático sobre uma janela.

- Sintaxe SAY:

@ 4,6 SAY "Código:" SIZE 70,7 PIXEL OF oTPanel1

- Sintaxe TSay(): TSay():New([anRow], [anCol], [abText], [aoWnd], [acPicture], [aoFont], [IPar7], [IPar8], [IPar9], [alPixels], [anClrText], [anClrBack], [anWidth], [anHeight], [IPar15], [IPar16], [IPar17], [IPar18], [IPar19])
- Parâmetros:

anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
abText	Code-Block, opcional. Quando executado deve retornar uma cadeia de caracteres a ser exibida.
aoWnd	Objeto, opcional. Janela ou diálogo onde o controle será criado.
acPicture	Caractere, opcional. Picture de formatação do conteúdo a ser exibido.
aoFont	Objeto, opcional. Objeto tipo tFont para configuração do tipo de fonte que será utilizado para exibir o conteúdo.
IPar7	Reservado.
IPar8	Reservado.
IPar9	Reservado.
alPixels	Lógico, opcional. Se .T. considera coordenadas passadas em pixels se .F., padrão, considera as coordenadas passadas em caracteres.
anClrText	Numérico, opcional. Cor do conteúdo do controle.
anClrBack	Numérico, opcional. Cor do fundo do controle.
anWidth	Numérico, opcional. Largura do controle em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
IPar15	Reservado.

IPar16	Reservado.
IPar17	Reservado.
IPar18	Reservado.
IPar19	Reservado.

Comando MSGET - Objeto: TGet()

O comando MsGet ou o objeto TGet é utilizado para criar um controle que armazene ou altere o conteúdo de uma variável através de digitação. O conteúdo da variável só é modificado quando o controle perde o foco de edição para outro controle.

- Sintaxe MSGET:

```
@ 3,192 MSGET dData PICTURE "99/99/99" SIZE 40,7 PIXEL OF oTPanel1
```

- Sintaxe TGet():New([anRow], [anCol], [abSetGet], [aoWnd], [anWidth], [anHeight], [acPict], [abValid], [anClrFore], [anClrBack], [aoFont], [IPar12], [oPar13], [alPixel], [cPar15], [IPar16], [abWhen], [IPar18], [IPar19], [abChange], [alReadOnly], [alPassword], [cPar23], [acReadVar], [cPar25], [IPar26], [nPar27], [IPar28])
- Parâmetros:

anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
abSetGet	Bloco de código, opcional. Bloco de código no formato { u IF(Pcount()>0, <var>:= u, <var>) } que o controle utiliza para atualizar a variável <var>. <var> deve ser tipo caracter, numérico ou data.
aoWnd	Objeto, opcional. Janela ou controle onde o controle será criado.
anWidth	Numérico, opcional. Largura do controle em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
acPict	Caractere, opcional. Máscara de formatação do conteúdo a ser exibido.
abValid	Bloco de código, opcional. Executado quando o conteúdo do controle deve ser validado, deve retornar .T. se o conteúdo for válido e .F. quando o conteúdo for inválido.
anClrFore	Numérico, opcional. Cor de fundo do controle.
anClrBack	Numérico, opcional. Cor do texto do controle.
aoFont	Objeto, opcional. Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do controle.
IPar12	Reservado.
oPar13	Reservado.
alPixel	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
cPar15	Reservado.
IPar16	Reservado.

abWhen	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
lPar18	Reservado.
lPar19	Reservado.
abChange	Bloco de código, opcional. Executado quando o controle modifica o valor da variável associada.
alReadOnly	Lógico, opcional. Se .T. o controle não poderá ser editado.
alPassword	Lógico, opcional. Se .T. o controle exibirá asteriscos "*" no lugar dos caracteres exibidos pelo controle para simular entrada de senha.
cPar23	Reservado.
acReadVar	Caractere, opcional. Nome da variável que o controle deverá manipular, deverá ser a mesma variável informada no parâmetro abSetGet, e será o retorno da função ReadVar().
cPar25	Reservado.
lPar26	Reservado.
nPar27	Reservado.
lPar18	Reservado.

Objeto MsGetDados()

Objeto tipo lista com uma ou mais colunas para cadastramento de dados baseado em um vetor. Sua utilização exige que seja utilizado três variáveis com seu escopo Private, são elas: aRotina, aHeader e aCOLS.

- Observações importantes:
 - O vetor aHeader deve ser construído com base no dicionário de dados.
 - O vetor aCOLS deve ser construído com base no vetor aHeader, porém deve-se criar uma coluna adicional para o controle de exclusão do registro, ou seja, quando o usuário acionar a tecla <DELETE> a linha ficará na cor cinza e esta coluna estará com o seu valor igual a verdadeiro (.T.).
 - Quando instanciado este objeto é possível saber em que linha o usuário está porque o objeto trabalha com uma variável de escopo Public denominada "n", seu valor é numérico e terá sempre no conteúdo a linha em que o usuário encontra-se com o cursor.
- Sintaxe: MSGETDADOS():NEW(nSuperior, nEsquerda, nInferior, nDireita, nOpc, [cLinhaOk], [cTudoOk], [cIniCpos], [lApagar], [aAlter], , [uPar1], [lVazio], [nMax], [cCampoOk], [cSuperApagar], [uPar2], [cApagaOk], [oWnd])
- Parâmetros:

nSuperior	Distancia entre a MsGetDados e o extremidade superior do objeto que a contém.
nEsquerda	Distancia entre a MsGetDados e o extremidade esquerda do objeto que a contém.
nInferior	Distancia entre a MsGetDados e o extremidade inferior do objeto que a contém.
nDireita	Distancia entre a MsGetDados e o extremidade direita do objeto que a contém.
nOpc	Posição do elemento do vetor aRotina que a MsGetDados usará como referencia.
cLinhaOk	Função executada para validar o contexto da linha atual do aCols.
cTudoOk	Função executada para validar o contexto geral da MsGetDados (todo aCols).
clniCpos	Nome dos campos do tipo caracter que utilizarão incremento automático. Este parâmetro deve ser no formato “+<nome do primeiro campo>+<nome do segundo campo>+...”.
lApagar	Habilita deletar linhas do aCols. Valor padrão falso.
aAlter	Vetor com os campos que poderão ser alterados.
uPar1	Parâmetro reservado.
lVazio	Habilita validação da primeira coluna do aCols para esta não poder estar vazia. Valor padrão falso.
nMax	Número máximo de linhas permitidas. Valor padrão 99.
cCampoOk	Função executada na validação do campo.
cSuperApagar	Função executada quando pressionada as teclas <Ctrl>+<Delete>.
uPar2	Parâmetro reservado.
cApagaOk	Função executada para validar a exclusão de uma linha do aCols.
oWnd	Objeto no qual a MsGetDados será criada.

Variável Private aRotina

Array com as rotinas que serão executadas na MBrowse e que definirá o tipo de operação que está sendo executada, por exemplo: Pesquisar, Visualizar, Incluir, Alterar, Excluir e outros.

Este vetor precisa ser construído no formato:

Elemento	Conteúdo
1	Título da opção.
2	Nome da rotina (Function).
3	Reservado.
4	Operação (1-Pesquisar;2-Visualizar;3-Incluir;4-Alterar;5-Exclusão).
5	Acesso relacionado a rotina, se está opção não for informada nenhum acesso será validado.

Variável Private aHeader

Array com informações das colunas, ou seja, com as características dos campos que estão contidas no dicionário de dados (SX3), este vetor precisa estar no formato a seguir:

Elemento	Conteúdo
1	Título do campo
2	Nome do campo
3	Máscara do campo
4	Tamanho do campo
5	Decimal do campo

Variável Private aCols

Vetor com as linhas a serem editadas. As colunas devem ser construídas com base no vetor aHeader e mais uma última coluna com o valor lógico que determina se a linha foi excluída, inicialmente esta deverá ter o seu conteúdo igual a falso (.F.).

1.4.2. Estrutura de um programa utilizando a Modelo2()

O exemplo abaixo demonstra a montagem de um programa para a utilização do protótipo Modelo 2. Antes de iniciarmos o exemplo vamos estruturar o programa.

Estrutura do programa

Linhas	Programa
1	Função principal;
2	Declaração e atribuição de variáveis;
3	Acesso a tabela principal e sua ordem;
4	Chamada da função MBrowse;
5	Fim da função principal.
6	
7	Função de visualização, alteração e exclusão;
8	Declaração e atribuição de variáveis;
9	Acesso ao primeiro registro da chave em que está posicionado na MBrowse;
10	Montagem das variáveis estáticas em tela;
11	Montagem do vetor aHeader por meio do dicionário de dados;
12	Montagem do vetor aCOLS de todos os registros referente a chave principal em que está posicionado na MBrowse;
13	Instância da MsDialog;

- 14 Instância dos objetos TSay e TGet;
- 15 Instância do objeto MsGetDados;
- 16 Ativar o objeto principal que é o objeto da janela;
- 17 Se for operação diferente de visualização e clicou no botão OK;
- 18 A operação é de Alteração?
- 19 Chamar a função para alterar os dados;
- 20 Caso contrário
- 21 Chamar a função para excluir os dados;
- 22 **Fim da função de visualização, alteração e exclusão.**
- 23
- 24 **Função de inclusão;**
- 25 Declaração e atribuição de variáveis;
- 26 Montagem das variáveis estáticas em tela;
- 27 Montagem do vetor aHeader por meio do dicionário de dados;
- 28 Montagem do vetor aCOLS com o seu conteúdo conforme o inicializador padrão do campo ou vazio, pois trata-se de uma inclusão;
- 29 Instância da MsDialog;
- 30 Instância dos objetos TSay e TGet;
- 31 Instância do objeto MsGetDados;
- 32 Ativar o objeto principal que é o objeto da janela;
- 33 Se clicou no botão OK;
- 34 Chamar a função para incluir os dados;
- 35 **Fim da função de inclusão.**

Rotina principal

```
#include "protheus.ch"

//+-----+
//| Rotina | xModelo2 | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Função exemplo do protótipo Modelo2. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
User Function xModelo2()
    Private cCadastro := "Protótipo Modelo 2"
```

```

Private aRotina := {}

AADD( aRotina, {"Pesquisar" , "AxPesqui" ,0,1})
AADD( aRotina, {"Visualizar" , 'U_Mod2Mnt',0,2})

AADD( aRotina, {"Incluir" , 'U_Mod2Inc',0,3})
AADD( aRotina, {"Alterar" , 'U_Mod2Mnt',0,4})
AADD( aRotina, {"Excluir" , 'U_Mod2Mnt',0,5})

dbSelectArea("ZA3")
dbSetOrder(1)
dbGoTop()

MBrowse(,,,, "ZA3")

Return

```

Rotina de inclusão

```

//+-----+
//| Rotina | Mod2Inc | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para incluir dados. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+

```

```

User Function Mod2Inc( cAlias, nReg, nOpc )
    Local oDlg
    Local oGet
    Local oTPanel1
    Local oTPanel2

    Local cCodigo := ZA3->ZA3_CODIGO
    Local cNome   := ZA3->ZA3_NOME
    Local dData   := dDataBase

    Private aHeader := {}
    Private aCOLS := {}

```

```

Private aREG := {}

dbSelectArea( cAlias )
dbSetOrder(1)

Mod2aHeader( cAlias )
Mod2aCOLS( cAlias, nReg, nOpc )

DEFINE MSDIALOG oDlg TITLE cCadastro From 8,0 To 28,80 OF oMainWnd

oTPanel1 := TPanel():New(0,0,"",oDlg,NIL,.T.,;
.F.,NIL,NIL,0,16,.T.,.F.)

oTPanel1:Align := CONTROL_ALIGN_TOP

@ 4, 006 SAY "Código:" SIZE 70,7 PIXEL OF
oTPanel1
@ 4, 062 SAY "Nome:" SIZE 70,7 PIXEL OF
oTPanel1
@ 4, 166 SAY "Emissao:" SIZE 70,7 PIXEL OF
oTPanel1

@ 3, 026 MSGET cCodigo F3 "SA3" PICTURE "@!"
VALID;
Mod2Vend(cCodigo, @cNome);

SIZE 030,7 PIXEL OF oTPanel1

@ 3, 080 MSGET cNome When .F. SIZE 78,7 PIXEL
OF oTPanel1

@ 3, 192 MSGET dData PICTURE "99/99/99" SIZE
40,7 PIXEL OF
oTPanel1

oTPanel2 := TPanel():New(0,0,"",oDlg,NIL,.T.,;
.F.,NIL,NIL,0,16,.T.,.F.)

oTPanel2:Align := CONTROL_ALIGN_BOTTOM

oGet :=

```

```
MSGetDados():New(0,0,0,0,nOpc,"U_Mod2LOk()";
".T.", "+ZA3_ITEM",.T.)
```

```
oGet:oBrowse:Align
```

```
:=
```

```
CONTROL_ALIGN_ALLCLIENT
```

```
    ACTIVATE MSDIALOG oDlg CENTER ON INIT ;
    EnchoiceBar(oDlg,{ | IIF(U_Mod2TOk(), Mod2Grvl(),;
( oDlg:End(), NIL ) )},{ | oDlg:End() })
```

```
Return
```

Rotina de Visualização, Alteração e Exclusão

```
//+-----+
//| Rotina | Mod2Mnt | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para Visualizar, Alterar e Excluir dados. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
```

```
User Function Mod2Mnt( cAlias, nReg, nOpc )
```

```
    Local oDlg
```

```
    Local oGet
```

```
    Local oTPanel1
```

```
    Local oTPanel2
```

```
    Local cCodigo := Space(Len(Space(ZA3->ZA3_CODIGO)))
```

```
    Local cNome := Space(Len(Space(ZA3->ZA3_NOME)))
```

```
    Local dData := Ctod(Space(8))
```

```
    Private aHeader := {}
```

```
    Private aCOLS := {}
```

```
    Private aREG := {}
```

```
    dbSelectArea( cAlias )
```

```
    dbGoTo( nReg )
```

```
cCodigo := ZA3->ZA3_CODIGO
```

```
cNome := ZA3->ZA3_NOME
```

```
cData := ZA3->ZA3_DATA
```

```
Mod2aHeader( cAlias )
```

```
Mod2aCOLS( cAlias, nReg, nOpc )
```

```
DEFINE MSDIALOG oDlg TITLE cCadastro From 8,0 To 28,80 OF oMainWnd
```

```

oTPane1 := TPanel():New(0,0,"",oDlg,NIL,.T.,;
.F.,NIL,NIL,0,16,.T.,.F.)

oTPane1:Align := CONTROL_ALIGN_TOP

@ 4, 006 SAY "Código:" SIZE 70,7 PIXEL OF
oTPane1

@ 4, 062 SAY "Nome:" SIZE 70,7 PIXEL OF
oTPane1

@ 4, 166 SAY "Emissao:" SIZE 70,7 PIXEL OF
oTPane1

@ 3, 026 MSGET cCodigo When .F. SIZE 30,7 PIXEL
OF oTPane1

@ 3, 080 MSGET cNome When .F. SIZE 78,7 PIXEL
OF oTPane1

@ 3, 192 MSGET dData When .F. SIZE 40,7 PIXEL
OF oTPane1

oTPanel2 := TPanel():New(0,0,"",oDlg,NIL,.T.,;
.F.,NIL,NIL,0,16,.T.,.F.)

oTPanel2:Align := CONTROL_ALIGN_BOTTOM

If nOpc == 4
    oGet :=
MSGetDados():New(0,0,0,0,nOpc,"U_Mod2LOk()";
".T.", "+ZA3_ITEM",.T.)
Else

```



```

                                oGet := MSGetDados():New(0,0,0,0,nOpc)
                                Endif
                                oGet:oBrowse:Align                               :=
CONTROL_ALIGN_ALLCLIENT

```

```

    ACTIVATE MSDIALOG oDlg CENTER ON INIT ;
    EnchoiceBar(oDlg,{ | ( IIF( nOpc==4, Mod2GrvA(), ;
    IIF( nOpc==5, Mod2GrvE(), oDlg:End() ) ), oDlg:End() ) },;
    { | oDlg:End() })
Return

```

Montagem do array aHeader

```

//+-----+
//| Rotina | Mod2aHeader | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para montar o vetor aHeader.          |
//+-----+
//| Uso   | Para treinamento e capacitação.              |
//+-----+
Static Function Mod2aHeader( cAlias )
    Local aArea := GetArea()
    dbSelectArea("SX3")
    dbSetOrder(1)
    dbSeek( cAlias )
    While !EOF() .And. X3_ARQUIVO == cAlias
        If X3Uso(X3_USADO) .And. cNivel >= X3_NIVEL
            AADD( aHeader, { Trim( X3Titulo() ),;
            X3_CAMPO;;
            X3_PICTURE;;
            X3_TAMANHO;;
            X3_DECIMAL;;
            X3_VALID;;
            X3_USADO;;
            X3_TIPO;;
            X3_ARQUIVO;;
            X3_CONTEXT})

```

```

Endif
dbSkip()

End
RestArea(aArea)

Return

```

Montagem do array aCols

```

//+-----+
//| Rotina | Mod2aCOLS | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para montar o vetor aCOLS.          |
//+-----+
//| Uso   | Para treinamento e capacitação.          |
//+-----+
Static Function Mod2aCOLS( cAlias, nReg, nOpc )
    Local aArea := GetArea()
    Local cChave := ZA3->ZA3_CODIGO
    Local nl := 0

    If nOpc <> 3

        dbSelectArea( cAlias )
        dbSetOrder(1)
        dbSeek( xFilial( cAlias ) + cChave )
        While !EOF() .And. ;
            ZA3->( ZA3_FILIAL + ZA3_CODIGO ) == xFilial(
cAlias ) + cChave

                AADD( aREG, ZA3->( RecNo() ) )
                AADD( aCOLS, Array( Len( aHeader ) + 1 ) )
                For nl := 1 To Len( aHeader )
                    If aHeader[nl,10] == "V"
                        aCOLS[Len(aCOLS),nl] :=

CriaVar(aHeader[nl,2],.T.)

                    Else
                        aCOLS[Len(aCOLS),nl] :=

FieldGet(FieldPos(aHeader[nl,2]))

                Endif
    Endif

```

```

Next nl
aCOLS[Len(aCOLS),Len(aHeader)+1] := .F.
dbSkip()
End

Else

AADD( aCOLS, Array( Len( aHeader ) + 1 ) )

For nl := 1 To Len( aHeader )
    aCOLS[1, nl] := CriaVar( aHeader[nl, 2], .T. )
Next nl
aCOLS[1, GdFieldPos("ZA3_ITEM")] := "01"
aCOLS[1, Len( aHeader )+1 ] := .F.

Endif
Restarea( aArea )
Return

```

Efetivação da inclusão

```

//+-----+
//| Rotina | Mod2GrvI | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para gravar os dados na inclusão. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
Static Function Mod2GrvI()
    Local aArea := GetArea()
    Local nl := 0
    Local nX := 0

    dbSelectArea("ZA3")
    dbSetOrder(1)
    For nl := 1 To Len( aCOLS )

        If ! aCOLS[nl,Len(aHeader)+1]
            RecLock("ZA3",.T.)
            ZA3->ZA3_FILIAL := xFilial("ZA3")
            ZA3->ZA3_CODIGO := cCodigo

```

```

                                ZA3->ZA3_DATA := dData
                                For nX := 1 To Len( aHeader )
                                    FieldPut( FieldPos( aHeader[nX, 2] ),
aCOLS[nI, nX] )

                                Next nX
                                MsUnlock()
                                Endif

                                Next nI

                                RestArea(aArea)
                                Return

```

Efetivação da alteração

```

//+-----+
//| Rotina | Mod2GrvA | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para gravar os dados na alteração.      |
//+-----+
//| Uso   | Para treinamento e capacitação.                |
//+-----+
Static Function Mod2GrvA()
    Local aArea := GetArea()
    Local nI := 0
    Local nX := 0

    dbSelectArea("ZA3")
    For nI := 1 To Len( aREG )

        If nI <= Len( aREG )
            dbGoTo( aREG[nI] )
            RecLock("ZA3",.F.)
            If aCOLS[nI, Len(aHeader)+1]
                dbDelete()
            Endif
        Else
            RecLock("ZA3",.T.)

```

```

Endif

If !aCOLS[nI, Len(aHeader)+1]
    ZA3->ZA3_FILIAL := xFilial("ZA3")
    ZA3->ZA3_CODIGO := cCodigo
    ZA3->ZA3_DATA := dData
    For nX := 1 To Len( aHeader )
        FieldPut( FieldPos( aHeader[nX, 2] ),

aCOLS[nI, nX] )

Next nX
Endif
MsUnLock()

Next nI
RestArea( aArea )
Return

```

Efetivação da exclusão

```

//+-----+
//| Rotina | Mod2GrvE | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para excluir os registros.      |
//+-----+
//| Uso   | Para treinamento e capacitação.      |
//+-----+
Static Function Mod2GrvE()
    Local nI := 0

    dbSelectArea("ZA3")
    For nI := 1 To Len( aCOLS )

        dbGoTo(aREG[nI])
        RecLock("ZA3",.F.)
        dbDelete()
        MsUnLock()
    
```

Next nl

Return

Função auxiliar: Validação do código do vendedor

```
//+-----+
//| Rotina | Mod2Vend | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para validar o código do vendedor.      |
//+-----+

//| Uso   | Para treinamento e capacitação.      |
//+-----+
```

```
Static Function Mod2Vend( cCodigo, cNome )
    If ExistCpo("SA3",cCodigo) .And. ExistChav("ZA3",cCodigo)
        cNome :=
    Posicione("SA3",1,xFilial("SA3")+cCodigo,"A3_NOME")
    Endif
    Return(!Empty(cNome))
```

Função auxiliar: Validação do código do centro de custo na mudança de linha

```
//+-----+
//| Rotina | Mod2LOk | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para validar a linha de dados.      |
//+-----+
//| Uso   | Para treinamento e capacitação.      |
//+-----+

User Function Mod2LOk()
Local lRet := .T.
Local cMensagem := "Não será permitido linhas sem o centro de custo."
    If !aCOLS[n, Len(aHeader)+1]
        If Empty(aCOLS[n,GdFieldPos("ZA3_CCUSTO")])
            MsgAlert(cMensagem,cCadastro)
            lRet := .F.
        Endif
    Endif
```

```

    Endif
Return( IRet )

```

Função auxiliar: Validação do código do centro de custo para todas as linhas

```

//+-----+
//| Rotina | Mod2TOk | Autor | Robson Luiz (rleg) | Data |01.01.2007 |
//+-----+
//| Descr. | Rotina para validar toda as linhas de dados.      |
//+-----+
//| Uso   | Para treinamento e capacitação.                  |
//+-----+

User Function Mod2TOk()
    Local IRet := .T.
    Local nl := 0
    Local cMensagem := "Não será permitido linhas sem o centro de custo."

    For nl := 1 To Len( aCOLS )
        If aCOLS[nl, Len(aHeader)+1]
            Loop
        Endif
        If !aCOLS[nl, Len(aHeader)+1]
            If Empty(aCOLS[n,GdFieldPos("ZA3_CCUSTO")])
                MsgAlert(cMensagem,cCadastro)
                IRet := .F.
            Exit
        Endif
    Endif

    Next nl
Return( IRet )

```

1.4.3. Função Modelo2()

A função Modelo2() é uma interface pré-definida pela Microsiga que implementa de forma padronizada os componentes necessários a manipulação de estruturas de dados nas quais o cabeçalho e os itens da informação compartilham o mesmo registro físico.

Seu objetivo é atuar como um facilitador de codificação, permitindo a utilização dos recursos básicos dos seguintes componentes visuais:

- MsDialog()
- TGet()
- TSay()
- MsNewGetDados()
- EnchoiceBar()

Importante

- A função Modelo2() não implementa as regras de visualização, inclusão, alteração e exclusão, como uma AxCadastro() ou AxFunction().
- A inicialização das variáveis Private utilizada nos cabeçalhos e rodapés, bem como a inicialização e gravação do aCols devem ser realizadas pela rotina que “suporta” a execução da Modelo2().
- Da mesma forma, o Browse deve ser tratado por esta rotina, sendo comum a Modelo2() estar vinculada ao uso de uma MBrowse().
- Sintaxe: Modelo2([cTitulo], [aCab], [aRoda], [aGrid], [nOpc], [cLinhaOk], [cTudoOk])
- Parâmetros:

cTitulo	Título da janela
aCab	Array contendo as informações que serão exibidas no cabeçalho na forma de Enchoice() aCab[n][1] (Caractere) := Nome da variável private que será vinculada ao campo da Enchoice(). aCab[n][2] (Array) := Array com as coordenadas do campo na tela {Linha, Coluna} aCab[n][3] (Caractere) := Título do campo na tela aCab[n][4] (Caractere) := Picture de formatação do get() do campo. aCab[n][5] (Caractere) := Função de validação do get() do campo. aCab[n][6] (Caractere) := Nome da consulta padrão que será executada para o campo via tecla F3 aCab[n][7] (Lógico) := Se o campo estará livre para digitação.
aRoda	Array contendo as informações que serão exibidas no cabeçalho na forma de Enchoice(), no mesmo formato que o aCab.
aGrid	Array contendo as coordenadas da GetDados() na tela. Padrão := {44,5,118,315}
nOpc	Opção selecionada na MBrowse, ou que deseje ser passada para controle da Modelo2, aonde: 2 – Visualizar 3 - Incluir 4 - Alterar 5 - Excluir

cLinhaOk	Função para validação da linha na GetDados()
cTudoOk	Função para validação na confirmação da tela de interface da Modelo2().

- Retorno:

Lógico	Indica se a tela da interface Modelo2() foi confirmada ou cancelada pelo usuário.
---------------	---

Exemplo: Utilização da Modelo2() para visualização do Cadastro de Tabelas (SX5)

```
#include "protheus.ch"
```

```
//+-----+
//| Rotina | MBRW2SX5| Autor | ARNALDO RAYMUNDO JR. | Data |01.01.2007 |
//+-----+
//| Descr. | UTILIZACAO DA MODELO2() PARA VISUALIZAÇÃO DO SX5.      |
//+-----+
//| Uso   | CURSO DE ADVPL          |
//+-----+
```

```
USER FUNCTION MBrw2Sx5()
```

```
Local cAlias                := "SX5"
```

```
Private cCadastro           := "Arquivo de Tabelas"
```

```
Private aRotina              := {}
```

```
Private cDelFunc             := ".T." // Validacao para a exclusao. Pode-se
utilizar ExecBlock
```

```
AADD(aRotina,{"Pesquisar"    ,"AxPesqui" ,0,1})
```

```
AADD(aRotina,{"Visualizar"   ,"U_SX52Vis" ,0,2})
```

```
AADD(aRotina,{"Incluir"      ,"U_SX52Inc" ,0,3})
```

```
AADD(aRotina,{"Alterar"      ,"U_SX52Alt" ,0,4})
```

```
AADD(aRotina,{"Excluir"      ,"U_SX52Exc" ,0,5})
```

```
dbSelectArea(cAlias)
```

```
dbSetOrder(1)
```

```
mBrowse( 6,1,22,75,cAlias)
```

```
Return
```

USER FUNCTION SX52INC(cAlias,nReg,nOpc)

```
//Local nUsado                := 0
Local cTitulo                  := "Inclusao de itens - Arquivo de Tabelas"

Local aCab                     := {} // Array com descricao dos campos do
Cabecalho do Modelo 2
Local aRoda                    := {} // Array com descricao dos campos do
Rodape do Modelo 2
Local aGrid                     := {80,005,050,300} //Array com coordenadas da
GetDados no modelo2 - Padrao: {44,5,118,315}
                                // Linha Inicial - Coluna Inicial - +Qts
Linhas - +Qts Colunas : {080,005,050,300}
Local cLinhaOk                 := "AlwaysTrue()" // Validacoes na linha da
GetDados da Modelo 2
Local cTudoOk                  := "AlwaysTrue()" // Validacao geral da GetDados
da Modelo 2
Local lRetMod2 := .F. // Retorno da função Modelo2 - .T. Confirmou / .F. Cancelou
Local nColuna                  := 0

// Variaveis para GetDados()
Private aCols                  := {}
Private aHeader                 := {}
```

Exemplo (continuação):

```
// Variaveis para campos da Enchoice()
Private cX5Filial := xFilial("SX5")
Private cX5Tabela := SPACE(5)

// Montagem do array de cabeçalho
// AADD(aCab,{"Variável"                ,{L,C} ,"Título","Picture","Valid","F3",lEnable})
AADD(aCab,{"cX5Filial"                  ,{015,010} ,"Filial","@!"",.F.})
AADD(aCab,{"cX5Tabela"                  ,{015,080} ,"Tabela","@!"",.T.})

// Montagem do aHeader
AADD(aHeader,{"Chave"                   , "X5_CHAVE","@!"",5,0,"AlwaysTrue()"},;
```

```

                                "", "C", "", "R"})
AADD(aHeader,{"Descricao"                                , "X5_DESCRI", "@!", 40, 0, "AlwaysTrue()";,
                                "", "C", "", "R"})

// Montagem do aCols
aCols := Array(1, Len(aHeader)+1)
// Inicialização do aCols
For nColuna := 1 to Len(aHeader)
    If aHeader[nColuna][8] == "C"
                                aCols[1][nColuna] := SPACE(aHeader[nColuna][4])
    ElseIf aHeader[nColuna][8] == "N"
                                aCols[1][nColuna] := 0
    ElseIf aHeader[nColuna][8] == "D"
                                aCols[1][nColuna] := CTOD("")
    ElseIf aHeader[nColuna][8] == "L"
                                aCols[1][nColuna] := .F.
    ElseIf aHeader[nColuna][8] == "M"
                                aCols[1][nColuna] := ""
    Endif

Next nColuna
aCols[1][Len(aHeader)+1] := .F. // Linha não deletada
IRetMod2 := Modelo2(cTitulo, aCab, aRoda, aGrid, nOpc, cLinhaOk, cTudoOk)
IF IRetMod2
    //MsgInfo("Você confirmou a operação", "MBRW2SX5")
    For nLinha := 1 to len(aCols)
                                // Campos de Cabeçalho
                                Reclock("SX5", .T.)
                                SX5->X5_FILIAL := cX5Filial
                                SX5->X5_TABELA := cX5Tabela
                                // Campos do aCols
                                //SX5->X5_CHAVE := aCols[nLinha][1]
                                //SX5->X5_DESCRI := aCols[nLinha][2]
                                For nColuna := 1 to Len(aHeader)
                                    SX5->&(aHeader[nColuna][2]) :=
aCols[nLinha][nColuna]
                                Next nColuna

```

MsUnLock()

Next nLinha

ELSE

MsgAlert("Você cancelou a operação","MBRW2SX5")

ENDIF

Return

Exercício

Implementar uma Modelo2() para a tabela padrão do ERP – SX5: Arquivo de Tabelas

Exercício

Implementar as funções de Visualização, Alteração e Exclusão para a Modelo2 desenvolvida para o SX5.

Exercício

Implementar uma Modelo2() para a tabela padrão do ERP – SB1: Tabela de Produtos.

1.5. Modelo3()

O nome Modelo 3, assim como a Modelo 2 foi conceituado pela Microsiga por se tratar de um protótipo de tela para entrada de dados. Inicialmente vamos desmistificar dois pontos:

- Função Modelo3() – Trata-se de uma função pronta que contempla o protótipo Modelo 3, porém, este é um assunto que não iremos tratar aqui, visto que é uma funcionalidade simples que quando necessário intervir em algo na rotina não há muito recurso para tal.
- Protótipo Modelo 3 – Trata-se de uma tela, como a figura abaixo, onde seu objetivo é efetuar a manutenção em vários registros de uma só vez relacionada a outro registro de outra tabela, ou seja, aqui teremos o relacionamento de registros “pai e filho”, então é preciso se preocupar com este relacionamento. Por exemplo: efetuar a manutenção em um pedido de vendas, onde terá um registro em uma tabela referente à cabeça do pedido e outra tabela com os registros referentes aos itens deste pedido de vendas.

Para ganharmos tempo não será apresentado aqui toda a explicação e montagens para a função EnchoiceBar, comando MsDialog, Say e MsGet e para os vetores aHeader e aCOLS, entretanto todos estes estarão na codificação do código fonte.

A tela de modelo 3 é constituído de MsDialog, EnchoiceBar, Enchoice, MsGetDados, Say e Get.

Diante dos expostos até o momento houve um novo nome para nós, é ele a função Enchoice, o que é?

Função Enchoice() – Objeto MsMGet()

A função Enchoice ou o objeto MsMGet são recursos baseados no dicionário de dados para verificar campos obrigatórios, validações, gatilhos, consulta padrão e etc. Assim também para criar pastas de cadastros. Estes podem ser usados tanto com variáveis de memórias com o escopo Private como diretamente os campos da tabela que se refere. A diferença entre a função Enchoice e o objeto MsMGet é que a função não retorna o nome da variável de objeto exportável criado.

A estrutura para montar um programa com o protótipo modelo 3 é semelhante ao protótipo modelo 2, porém a diferença real é a utilização da função Enchoice ou o objeto MsMGet, para este documento iremos trabalhar com a função.

- Sintaxe: Enchoice(cAlias, nReg, nOpc, aAc, cOpc, cTextExclui, aAcho, aPos, aCpos, nNum, nColMens, cMensagem, cTudOk, oObj, lVirtual)
- Parâmetros:

cAlias	Alias do dados a serem cadastrados.
nReg	Número do registro da tabela a ser editado.
uPar1	Parâmetro reservado.
uPar2	Parâmetro reservado.
uPar3	Parâmetro reservado.
aAcho	Vetor com os campos que serão apresentados pela MsMGet.
aPos	Vetor com as coordenadas onde a MsMGet será criada no formato {coord. superior, coord. esquerda, coord. direita, coord. inferior}. Função executada para validar o contexto da linha atual do aCols.
aCpos	Vetor com os campos que poderão ser alterados.
uPar4	Parâmetro reservado. Nome dos campos do tipo caracter que utilizarão incremento automático. Este parâmetro deve ser no formato "+<nome do primeiro campo>+<nome do segundo campo>+..."
uPar5	Parâmetro reservado.
uPar6	Parâmetro reservado.
uPar7	Parâmetro reservado.
oWnd	Objeto no qual a MsMGet será criada.
uPar8	Parâmetro reservado.
lMemoria	Indica se será usado variáveis de memória ou os campos da tabela para cadastramento dos dados. Valor padrão falso.
lColuna	Indica se a MsMGet será apresentada com um objeto por linha (uma coluna). Valor padrão falso. Parâmetro reservado.
uPar9	Parâmetro reservado.
lSemPastas	Indica se não será usado as Pastas de Cadastro na MsMGet. Função executada para validar a exclusão de uma linha do aCols.

Vale lembrar que nós programadores reaproveitamos muito o que já existe, isto é para simplesmente ganharmos tempo, e no caso da utilização da função Enchoice é preciso criar as variáveis de memórias que levam o mesmo nome dos campos da tabela em questão. Por exemplo o campo A2_NOME da tabela SA2 (cadastro de fornecedores) quando queremos referenciar o campo usa-se o prefixo da tabela e o campo em questão, desta forma:

SA2->A2_NOME

Agora quando queremos referenciar a uma variável que está com o conteúdo do mesmo campo criamos outro recurso, desta forma:

M->A2_NOME

E para criar variáveis com o nome do campo utilizamos um código de bloco (code-block) e mais um laço de leitura para atribuir valores iniciais a cada uma dela. Então fica assim o procedimento:

```
Private bCampo := { |nField| Field(nField) }
```

E em outro momento aproveitamos esta variável bCampo para facilitar a atribuição, veja o exemplo abaixo :

```
For nX := 1 To FCount()
M->&( Eval( bCampo, nX ) ) := Atribuição inicial ou atribuição de valor
Next nX
```

Ou seja, fazer para todos os campos, e a cada campo criar a variável com a atribuição inicial ou atribuição de valor.

1.5.1. Estrutura de um programa utilizando a Modelo3()

O exemplo abaixo demonstra a montagem de um programa para a utilização do protótipo Modelo 3. Antes de iniciarmos o exemplo vamos estruturar o programa.

Estrutura do programa

Linhas	Programa
1	Função principal;
2	Declaração e atribuição de variáveis;
3	Acesso a tabela principal e sua ordem;
4	Chamada da função MBrowse;
5	Fim da função principal.
6	
7	Função de visualização, alteração e exclusão;
8	Declaração e atribuição de variáveis;

```

9      Acesso ao primeiro registro da chave em que está posicionado na MBrowse;
10     Construção das variáveis de memória M->???;
11     Montagem do vetor aHeader por meio do dicionário de dados;
12     Montagem do vetor aCOLS de todos os registros referente a chave principal em
        que está posicionado na MBrowse;
13     Instância da MsDialog;
14     Execução da função Enchoice;
15     Instância do objeto MsGetDados;
16     Ativar o objeto principal que é o objeto da janela;
17     Se for operação diferente de visualização e clicou no botão OK;
18         A operação e de Alteração?
19             Chamar a função para alterar os dados;
20             Caso contrário
21                 Chamar a função para excluir os dados;
22     Fim da função de visualização, alteração e exclusão.
23
24     Função de inclusão;
25     Declaração e atribuição de variáveis;
26     Construção das variáveis de memória M->???;
27     Montagem do vetor aHeader por meio do dicionário de dados;
28     Montagem do vetor aCOLS com o seu conteúdo conforme o inicializador padrão
        do campo ou vazio, pois trata-se de uma inclusão;
29     Instância da MsDialog;
30     Instância dos objetos TSay e TGet;
31     Instância do objeto MsGetDados;
32     Ativar o objeto principal que é o objeto da janela;
33     Se clicou no botão OK;
34         Chamar a função para incluir os dados;
35     Fim da função de inclusão.

```

Rotina principal

```

//+-----+
//| Rotina | xModelo3 | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Função exemplo do protótipo Modelo3.          |
//+-----+

```

```
//| Uso | Para treinamento e capacitação. |

//+-----+
#include "Protheus.ch"

User Function xModelo3()
Private cCadastro := "Protótipo Modelo 3"
Private aRotina := {}
Private oCliente
Private oTotal
Private cCliente := ""
Private nTotal := 0

Private bCampo := { | nField | FieldName(nField) }

Private aSize := {}
Private aInfo := {}
Private aObj := {}
Private aPObj := {}
Private aPGet := {}

// Retorna a área útil das janelas Protheus
aSize := MsAdvSize()

// Será utilizado três áreas na janela
// 1ª - Enchoice, sendo 80 pontos pixel
// 2ª - MsGetDados, o que sobrar em pontos pixel é para este objeto
// 3ª - Rodapé que é a própria janela, sendo 15 pontos pixel
AADD( aObj, { 100, 080, .T., .F. })
AADD( aObj, { 100, 100, .T., .T. })
AADD( aObj, { 100, 015, .T., .F. })

// Cálculo automático da dimensões dos objetos (altura/largura) em pixel
aInfo := { aSize[1], aSize[2], aSize[3], aSize[4], 3, 3 }
aPObj := MsObjSize( aInfo, aObj )

// Cálculo automático de dimensões dos objetos MSGET
```



```
aPGet := MsObjGetPos( aSize[3] - aSize[1]), 315, { {004, 024, 240, 270} } )
```

```
AADD( aRotina, {"Pesquisar" , "AxPesqui" ,0,1})
AADD( aRotina, {"Visualizar" , 'U_Mod3Mnt',0,2})
AADD( aRotina, {"Incluir" , 'U_Mod3Inc',0,3})
AADD( aRotina, {"Alterar" , 'U_Mod3Mnt',0,4})
AADD( aRotina, {"Excluir" , 'U_Mod3Mnt',0,5})
```

```
dbSelectArea("ZA1")
dbSetOrder(1)
dbGoTop()
MBrowse(,,,"ZA1")
Return
```

Função de Inclusão

```
//+-----+
//| Rotina | Mod3Inc | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para incluir dados. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
User Function Mod3Inc( cAlias, nReg, nOpc )
Local oDlg
Local oGet
Local nX := 0
Local nOpcA := 0

Private aHeader := {}
Private aCOLS := {}
Private aGets := {}
Private aTela := {}

dbSelectArea( cAlias )
dbSetOrder(1)
```

```
For nX := 1 To FCount()
    M->&( Eval( bCampo, nX ) ) := CriaVar( FieldName( nX ), .T. )

Next nX

Mod3aHeader()
Mod3aCOLS( nOpc )

DEFINE MSDIALOG oDlg TITLE cCadastro FROM ;
aSize[7],aSize[1] TO aSize[6],aSize[5] OF oMainWnd PIXEL
    EnChoice( cAlias, nReg, nOpc, , , , aPObj[1])

    // Atualização do nome do cliente
    @ aPObj[3,1],aPGet[1,1] SAY "Cliente: " SIZE 70,7 OF oDlg PIXEL
    @ aPObj[3,1],aPGet[1,2] SAY oCliente VAR cCliente SIZE 98,7 OF oDlg PIXEL

    // Atualização do total
    @ aPObj[3,1],aPGet[1,3] SAY "Valor Total: " SIZE 70,7 OF oDlg PIXEL
    @ aPObj[3,1],aPGet[1,4] SAY oTotal VAR nTotal ;
    PICT "@E 9,999,999,999.99" SIZE 70,7 OF oDlg PIXEL

    oGet := MSGetDados():New(aPObj[2,1],aPObj[2,2],aPObj[2,3],aPObj[2,4],;
    nOpc,"U_Mod3LOk()",".T.", "+ZA2_ITEM",.T.)

ACTIVATE MSDIALOG oDlg ON INIT EnchoiceBar(oDlg,;
{| | IIF( Mod3TOk().And.Obrigatorio( aGets, aTela ), ( nOpcA := 1, oDlg:End() ), NIL ) },;
{| | oDlg:End() })

If nOpcA == 1 .And. nOpc == 3
    Mod3Grv( nOpc )
    ConfirmSXE()
Endif
Return
```

Função de Visualização, Alteração e Exclusão

```
//+-----+
//| Rotina | Mod3Mnt | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+

//| Descr. | Rotina para Visualizar, Alterar e Excluir dados. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+

User Function Mod3Mnt( cAlias, nReg, nOpc )
Local oDlg
Local oGet
Local nX := 0
Local nOpcA := 0
Private aHeader := {}
Private aCOLS := {}
Private aGets := {}
Private aTela := {}
Private aREG := {}

dbSelectArea( cAlias )
dbSetOrder(1)

For nX := 1 To FCount()
    M->&( Eval( bCampo, nX ) ) := FieldGet( nX )
Next nX

Mod3aHeader()
Mod3aCOLS( nOpc )
DEFINE MSDIALOG oDlg TITLE cCadastro FROM ;
aSize[7],aSize[1] TO aSize[6],aSize[5] OF oMainWnd PIXEL
    EnChoice( cAlias, nReg, nOpc, , , , aPObj[1])

    // Atualização do nome do cliente
    @ aPObj[3,1],aPGet[1,1] SAY "Cliente: " SIZE 70,7 OF oDlg PIXEL
    @ aPObj[3,1],aPGet[1,2] SAY oCliente VAR cCliente SIZE 98,7 OF oDlg PIXEL
```

```
// Atualização do total
@ aPObj[3,1],aPGet[1,3] SAY "Valor Total: " SIZE 70,7 OF oDlg PIXEL
@ aPObj[3,1],aPGet[1,4] SAY oTotal VAR nTotal PICTURE ;
"@E 9,999,999,999.99" SIZE 70,7 OF oDlg PIXEL

U_Mod3Cli()

oGet := MSGetDados():New(aPObj[2,1],aPObj[2,2],aPObj[2,3],aPObj[2,4],;
nOpc,"U_Mod3LOk()",".T.", "+ZA2_ITEM",.T.)

ACTIVATE MSDIALOG oDlg ON INIT EnchoiceBar(oDlg,;
{| | IIF( Mod3TOK().And.Obrigatorio( aGets, aTela ), ( nOpcA := 1, oDlg:End() ), NIL ) },;
{| | oDlg:End() })

If nOpcA == 1 .And. ( nOpc == 4 .Or. nOpc == 5 )
Mod3Grv( nOpc, aREG )
Endif
Return
```

Função para montar o vetor aHeader

```
//+-----+
//| Rotina | Mod3aHeader | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para montar o vetor aHeader. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
Static Function Mod3aHeader()
Local aArea := GetArea()

dbSelectArea("SX3")
dbSetOrder(1)
dbSeek("ZA2")
While !EOF() .And. X3_ARQUIVO == "ZA2"
    If X3Uso(X3_USADO) .And. cNivel >= X3_NIVEL
```

```

                                AADD( aHeader, { Trim( X3Titulo() ),;
                                X3_CAMPO;;
                                X3_PICTURE;;
                                X3_TAMANHO;;
                                X3_DECIMAL;;
                                X3_VALID;;
                                X3_USADO;;
                                X3_TIPO;;
                                X3_ARQUIVO;;
                                X3_CONTEXT})

                                Endif
                                dbSkip()

                                End
                                RestArea(aArea)
                                Return

```

Função para montar o vetor aCols

```

//+-----+
//| Rotina | Mod3aCOLS | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para montar o vetor aCOLS.          |
//+-----+
//| Uso   | Para treinamento e capacitação.          |
//+-----+
Static Function Mod3aCOLS( nOpc )
Local aArea := GetArea()
Local cChave := ""
Local cAlias := "ZA2"
Local nl := 0

If nOpc <> 3
    cChave := ZA1->ZA1_NUM

    dbSelectArea( cAlias )
    dbSetOrder(1)
    dbSeek( xFilial( cAlias ) + cChave )

```

```

While !EOF() .And. ZA2->( ZA2_FILIAL + ZA2_NUM ) == xFilial( cAlias ) + cChave
    AADD( aREG, ZA2->( RecNo() ) )
    AADD( aCOLS, Array( Len( aHeader ) + 1 ) )
    For nl := 1 To Len( aHeader )
        If aHeader[nl,10] == "V"
            aCOLS[Len(aCOLS),nl] :=

CriaVar(aHeader[nl,2],.T.)

            Else
                aCOLS[Len(aCOLS),nl] :=

FieldGet(FieldPos(aHeader[nl,2]))

            Endif
        Next nl
        aCOLS[Len(aCOLS),Len(aHeader)+1] := .F.
        dbSkip()
    End
Else
    AADD( aCOLS, Array( Len( aHeader ) + 1 ) )
    For nl := 1 To Len( aHeader )
        aCOLS[1, nl] := CriaVar( aHeader[nl, 2], .T. )

    Next nl
    aCOLS[1, GdFieldPos("ZA2_ITEM")] := "01"
    aCOLS[1, Len( aHeader )+1 ] := .F.
Endif
Restarea( aArea )
Return

```

Função para atribuir o nome do cliente a variável

```

//+-----+
//| Rotina | Mod3Cli | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para atualizar a variável com o nome do cliente. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+

```

```
User Function Mod3Cli()
cCliente := Posicione( "SA1", 1, xFilial("SA1") + M->(ZA1_CLIENT + ZA1_LOJA), "A1_NREDUZ" )
oCliente:Refresh()
Return(.T.)
```

Função para validar a mudança de linha na MsGetDados()

```
//+-----+
//| Rotina | Mod3LOk | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para atualizar a variável com o total dos itens. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
User Function Mod3LOk()
Local nl := 0
nTotal := 0
For nl := 1 To Len( aCOLS )
    If aCOLS[nl,Len(aHeader)+1]
        Loop
    Endif
    nTotal+=Round(aCOLS[nl,GdFieldPos("ZA2_QTDVEN")]*;
    aCOLS[nl,GdFieldPos("ZA2_PRCVEN")],2)
Next nl
oTotal:Refresh()
Return(.T.)
```

Função para validar se todas as linhas estão preenchidas

```
//+-----+
//| Rotina | Mod3TOk | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
//+-----+
//| Descr. | Rotina para validar os itens se foram preenchidos. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
Static Function Mod3TOk()
Local nl := 0
```

```
Local lRet := .T.
```

```
For nl := 1 To Len(aCOLS)
```

```
  If aCOLS[nl, Len(aHeader)+1]
```

```
    Loop
```

```
  Endif
```

```
  If Empty(aCOLS[nl, GdFieldPos("ZA2_PRODUT")]) .And. lRet
```

```
    MsgAlert("Campo PRODUTO preenchimento obrigatorio", cCadastro)
```

```
    lRet := .F.
```

```
  Endif
```

```
  If Empty(aCOLS[nl, GdFieldPos("ZA2_QTDVEN")]) .And. lRet
```

```
    MsgAlert("Campo QUANTIDADE preenchimento obrigatorio", cCadastro)
```

```
    lRet := .F.
```

```
  Endif
```

```
  If Empty(aCOLS[nl, GdFieldPos("ZA2_PRCVEN")]) .And. lRet
```

```
    MsgAlert("Campo PRECO UNITARIO preenchimento obrigatorio", cCadastro)
```

```
    lRet := .F.
```

```
  Endif
```

```
  If !lRet
```

```
    Exit
```

```
  Endif
```

```
Next i
```

```
Return( lRet )
```

Função para efetuar a gravação dos dados em ZA1 e ZA2 na inclusão, alteração e exclusão.

```
//+-----+
```

```
//| Rotina | Mod3Grv | Autor | Robson Luiz (rleg) | Data | 01.01.2007 |
```

```
//+-----+
```

```
//| Descr. | Rotina para efetuar a gravação nas tabelas. |
```

```
//+-----+
```

```
//| Uso | Para treinamento e capacitação. |
```

```
//+-----+
```

```
Static Function Mod3Grv( nOpc, aAlterar )
```

```
Local nX := 0
```



```
Local nl := 0
```

```
// Se for inclusão
```

```
If nOpc == 3
```

```
    // Grava os itens
```

```
    dbSelectArea("ZA2")
```

```
    dbSetOrder(1)
```

```
    For nX := 1 To Len( aCOLS )
```

```
        If !aCOLS[ nX, Len( aCOLS ) + 1 ]
```

```
            RecLock( "ZA2", .T. )
```

```
            For nl := 1 To Len( aHeader )
```

```
                FieldPut( FieldPos( Trim( aHeader[nl, 2] )
```

```
            ), aCOLS[nX,nl] )
```

```
            Next nl
```

```
            ZA2->ZA2_FILIAL := xFilial("ZA2")
```

```
            ZA2->ZA2_NUM := M->ZA1_NUM
```

```
            MsUnLock()
```

```
        Endif
```

```
    Next nX
```

```
    // Grava o Cabeçalho
```

```
    dbSelectArea( "ZA1" )
```

```
    RecLock( "ZA1", .T. )
```

```
    For nX := 1 To FCount()
```

```
        If "FILIAL" $ FieldName( nX )
```

```
            FieldPut( nX, xFilial( "ZA1" ) )
```

```
        Else
```

```
            FieldPut( nX, M->&( Eval( bCampo, nX ) ) )
```

```
        Endif
```

```
    Next nX
```

```
    MsUnLock()
```

```
Endif
```

```
// Se for alteração
```

```
If nOpc == 4
```

```
    // Grava os itens conforme as alterações
```

```

dbSelectArea("ZA2")
dbSetOrder(1)
For nX := 1 To Len( aCOLS )

    If nX <= Len( aREG )
        dbGoto( aREG[nX] )
        RecLock("ZA2",.F.)
        If aCOLS[ nX, Len( aHeader ) + 1 ]

            dbDelete()
        Endif
    Else

        If !aCOLS[ nX, Len( aHeader ) + 1 ]
            RecLock( "ZA2", .T. )
        Endif
    Endif

    If !aCOLS[ nX, Len(aHeader)+1 ]
        For nI := 1 To Len( aHeader )
            FieldPut( FieldPos( Trim( aHeader[ nI, 2 ] ) ),;
                aCOLS[ nX, nI ] )
        Next nI
        ZA2->ZA2_FILIAL := xFilial("ZA2")
        ZA2->ZA2_NUM := M->ZA1_NUM
    Endif
    MsUnLock()

Next nX

// Grava o Cabeçalho
dbSelectArea("ZA1")
RecLock( "ZA1", .F. )
For nX := 1 To FCount()

    If "FILIAL" $ FieldName( nX )
        FieldPut( nX, xFilial("ZA1"))
    Else
        FieldPut( nX, M->&( Eval( bCampo, nX ) ) )
    Endif

```

```

        Next
        MsUnlock()
    Endif

    // Se for exclusão
    If nOpc == 5
        // Deleta os Itens
        dbSelectArea("ZA2")

        dbSetOrder(1)
        dbSeek(xFilial("ZA2") + M->ZA1_NUM)
        While !EOF() .And. ZA2->(ZA2_FILIAL + ZA2_NUM) == xFilial("ZA2") +;
M->ZA1_NUM
            RecLock("ZA2")
            dbDelete()
            MsUnlock()
            dbSkip()

        End
        // Deleta o Cabeçalho
        dbSelectArea("ZA1")
        RecLock("ZA1",.F.)
        dbDelete()
        MsUnlock()
    Endif
    Return

```

1.5.2. Função Modelo3()

A função Modelo3) é uma interface pré-definida pela Microsiga que implementa de forma padronizada os componentes necessários a manipulação de estruturas de dados nas quais o cabeçalho e os itens da informação estão em tabelas separadas.

Seu objetivo é atuar como um facilitador de codificação, permitindo a utilização dos recursos básicos dos seguintes componentes visuais:

- MsDialog()
- Enchoice()
- EnchoiceBar()
- MsNewGetDados()

Importante

- A função Modelo3() não implementa as regras de visualização, inclusão, alteração e exclusão, como uma AxCadastro() ou AxFunction().
- A inicialização dos campos utilizados na Enchoice() deve ser realizadas pela rotina que “suporta” a execução da Modelo3(), normalmente através do uso da função RegToMemory().
- Da mesma forma, o Browse deve ser tratado por esta rotina, sendo comum a Modelo3() estar vinculada ao uso de uma MBrowse().
 - Sintaxe: Modelo3 ([cTitulo], [cAliasE], [cAliasGetD], [aCposE], [cLinOk], [cTudOk], [nOpcE], [nOpcG], [cFieldOk])
 - Parâmetros:

cTitulo	Título da janela
cAliasE	Alias da tabela que será utilizada na Enchoice
cAliasGetD	Alias da tabela que será utilizada na GetDados
aCposE	Nome dos campos, pertencentes ao Alias especificado o parâmetro cAliasE, que deverão ser exibidos na Enchoice: AADD(aCposE,{"nome_campo"})
cLinhaOk	Função para validação da linha na GetDados()
cTudoOk	Função para validação na confirmação da tela de interface da Modelo2().
nOpcE	Opção selecionada na MBrowse, ou que deseje ser passada para controle da Enchoice da Modelo3, aonde: 2 – Visualizar 3 - Incluir 4 - Alterar 5 - Excluir
nOpcG	Opção selecionada na MBrowse, ou que deseje ser passada para controle da GetDados da Modelo3, aonde: 2 – Visualizar 3 - Incluir 4 - Alterar 5 - Excluir
cFieldOk	Validação dos campos da Enchoice()

- Retorno:

Lógico	Indica se a tela da interface Modelo2() foi confirmada ou cancelada pelo usuário.
---------------	---

Exemplo: Utilização da Modelo3() para Pedidos de Vendas (SC5,SC6)

```
#INCLUDE "protheus.ch"
```

```
//+-----+
//| Rotina | MBRWMOD3| Autor | ARNALDO RAYMUNDO JR. |Data | 01.01.2007 |
//+-----+
//| Descr. | EXEMPLO DE UTILIZACAO DA MODELO3().      |
//+-----+
//| Uso   | CURSO DE ADVPL                          |
//+-----+
```

```
User Function MbrwMod3()
```

```
Private cCadastro           := "Pedidos de Venda"
Private aRotina             := {}
Private cDelFunc            := ".T." // Validacao para a exclusao. Pode-se
utilizar ExecBlock
Private cAlias               := "SC5"
```

```
AADD(aRotina,{ "Pesquisa","AxPesqui" ,0,1})
AADD(aRotina,{ "Visual" , "U_Mod3All" ,0,2})
AADD(aRotina,{ "Inclui"      ,"U_Mod3All" ,0,3})
AADD(aRotina,{ "Altera"     ,"U_Mod3All" ,0,4})
AADD(aRotina,{ "Exclui"     ,"U_Mod3All" ,0,5})
```

```
dbSelectArea(cAlias)
dbSetOrder(1)
mBrowse( 6,1,22,75,cAlias)
```

```
Return
```

```
User Function Mod3All(cAlias,nReg,nOpcx)
```

```
Local cTitulo := "Cadastro de Pedidos de Venda"
Local cAliasE := "SC5"
Local cAliasG := "SC6"
```



```

Aadd(aHeader,{ TRIM(x3_titulo), x3_campo, x3_picture,;
              x3_tamanho, x3_decimal,"AllwaysTrue()";
              x3_usado, x3_tipo, x3_arquivo, x3_context } )
Endif
dbSkip()
End

If nOpcx==3 // Incluir
    aCols:=Array(nUsado+1)
    aCols[1,nUsado+1]:=F.
    For nX:=1 to nUsado

                                                aCols[1,nX]:=CriaVar(aHeader[nX,2])

    Next
Else
    aCols:={}
    dbSelectArea("SC6")
    dbSetOrder(1)
    dbSeek(xFilial()+M->C5_NUM)
    While !eof().and.C6_NUM==M->C5_NUM
                                                AADD(aCols,Array(nUsado+1))
                                                For nX:=1 to nUsado

aCols[Len(aCols),nX]:=FieldGet(FieldPos(aHeader[nX,2]))
                                                Next
                                                aCols[Len(aCols),nUsado+1]:=F.
                                                dbSkip()

    End
Endif

If Len(aCols)>0
    //ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÿ
    //³ Executa a Modelo 3                                     3
    //ÀAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÙ
    aCposE := {"C5_CLIENTE"}

    lRetMod3 := Modelo3(cTitulo, cAliasE, cAliasG, aCposE, cLinOk, cTudOk,;

```

```

                                nOpcE, nOpcG,cFieldOk)
//ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA¿
//³ Executar processamento                                     3
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÙ
If lRetMod3
                                Aviso("Modelo3()", "Confirmada
operacao!", {"Ok"})
    Endif
Endif

Return

```

Exercício

Implementar uma Modelo3() para as tabelas padrões do ERP – SF1: Cab. NFE e SD1: Itens NFE.

2. Relatórios não gráficos

Os relatórios desenvolvidos em ADVPL possuem um padrão de desenvolvimento que mais depende de layout e tipos de parâmetros do que qualquer outro tipo de informação, visto que até o momento percebemos que a linguagem padrão da Microsiga é muito mais composta de funções genéricas do que de comandos.

Este tipo de relatório é caracterizado por um formato de impressão tipo PostScript®, e permite a geração de um arquivo em formato texto (.txt), com uma extensão própria da aplicação ERP (##R).

A estrutura de um relatório não gráfico é baseada no uso da função SetPrint(), complementada pelo uso de outras funções acessórias, as quais estão detalhadas no Guia de Referência Rápida que acompanha este material.

2.1. Funções Utilizadas para Desenvolvimento de Relatórios

2.1.1. SetPrint()

A função que cria a interface com as opções configuração para impressão de um relatório no formato texto. Basicamente duas variáveis m_pag e aReturn precisam ser declaradas como privadas (private) antes de executar a SetPrint(). Após confirmada, os dados são armazenados no vetor aReturn que será passado como parâmetro para função SetDefault().

- Sintaxe: SetPrint (< cAlias > , < cProgram > , [cPergunte] , [cTitle] , [cDesc1] , [cDesc2] , [cDesc3] , [IDic] , [aOrd] , [ICompres] , [cSize] , [uParm12] , [IFilter] , [ICrystal] , [cNameDrv] , [uParm16] , [IServer] , [cPortPrint]) --> cReturn
- Parâmetros:

cAlias	Alias do arquivo a ser impresso.
cProgram	Nome do arquivo a ser gerado em disco.
cPergunte	Grupo de perguntas cadastrado no dicionário SX1.
cTitle	Título do relatório.
cDesc1	Descrição do relatório.
cDesc2	Continuação da descrição do relatório.
cDesc3	Continuação da descrição do relatório.
IDic	Utilizado na impressão de cadastro genérico permite a escolha dos campos a serem impressos. Se o parametro cAlias não for informado o valor padrão assumido será .F.
aOrd	Ordem(s) de impressão.
ICompres	Se verdadeiro (.T.) permite escolher o formato da impressão, o valor padrão assumido será .T.
cSize	Tamanho do relatório "P", "M" ou "G".

uParm12	Parâmetro reservado
lFilter	Se verdadeiro (.T.) permite a utilização do assistente de filtro, o valor padrão assumido será .T.
lCrystal	Se verdadeiro (.T.) permite integração com Crystal Report, o valor padrão assumido será .F.
cNameDrv	Nome de um driver de impressão.
uParm16	Parâmetro reservado.
lServer	Se verdadeiro (.T.) força impressão no servidor.
cPortPrint	Define uma porta de impressão padrão.

- Retorno:

Caracter	Nome do Relatório
-----------------	-------------------

- Estrutura aReturn:

aReturn[1]	Caracter, tipo do formulário
aReturn[2]	Numérico, opção de margem
aReturn[3]	Caracter, destinatário
aReturn[4]	Numérico, formato da impressão
aReturn[5]	Numérico, dispositivo de impressão
aReturn[6]	Caracter, driver do dispositivo de impressão
aReturn[7]	Caracter, filtro definido pelo usuário
aReturn[8]	Numérico, ordem
aReturn[x]	A partir a posição [9] devem ser informados os nomes dos campos que devem ser considerados no processamento, definidos pelo uso da opção Dicionário da SetPrint().

2.1.2. SetDefault()

A função SetDefault() prepara o ambiente de impressão de acordo com as informações configuradas no array aReturn, obtidas através da função SetPrint().

- Sintaxe: SetDefault (< aReturn > , < cAlias > , [uParm3] , [uParm4] , [cSize] , [nFormat])
- Parâmetros:

aReturn	Configurações de impressão.
cAlias	Alias do arquivo a ser impresso.
uParm3	Parâmetro reservado.
uParm4	Parâmetro reservado.
cSize	Tamanho da página "P", "M" ou "G"
nFormat	Formato da página, 1 retrato e 2 paisagem.

- Estrutura aReturn:

aReturn[1]	Caracter, tipo do formulário
aReturn[2]	Numérico, opção de margem
aReturn[3]	Caracter, destinatário
aReturn[4]	Numérico, formato da impressão
aReturn[5]	Numérico, dispositivo de impressão
aReturn[6]	Caracter, driver do dispositivo de impressão
aReturn[7]	Caracter, filtro definido pelo usuário
aReturn[8]	Numérico, ordem
aReturn[x]	A partir a posição [9] devem ser informados os nomes dos campos que devem ser considerados no processamento, definidos pelo uso da opção Dicionário da SetPrint().

2.1.3. RptStatus()

Régua de processamento simples, com apenas um indicador de progresso, utilizada no processamento de relatórios do padrão SetPrint().

- Sintaxe: RptStatus(bAcao, cMensagem)
- Parâmetros:

bAcao	Bloco de código que especifica a ação que será executada com o acompanhamento da régua de processamento.
cMensagem	Mensagem que será exibida na régua de processamento durante a execução.

2.1.3.1. SETREGUA()

A função SetRegua() é utilizada para definir o valor máximo da régua de progressão criada através da função RptStatus().

- Sintaxe: SetRegua(nMaxProc)
- Parâmetros:

nMaxProc	Variável que indica o valor máximo de processamento (passos) que serão indicados pela régua.
-----------------	--

2.1.3.2. INCREGUA()

A função IncRegua() é utilizada para incrementar valor na régua de progressão criada através da função RptStatus()

- Sintaxe: IncRegua(cMensagem)
- Parâmetros:

cMensagem	Mensagem que será exibida e atualizada na régua de processamento a cada execução da função IncRegua(), sendo que a taxa de atualização da interface é controlada pelo Binário.
------------------	--

2.1.4. CABEC()

A função CABEC() determina as configurações de impressão do relatório e imprime o cabeçalho do mesmo.

- Sintaxe: Cabec(cTitulo, cCabec1, cCabec2, cNomeProg, nTamanho, nCompress, aCustomText, lPerg, cLogo)
- Parâmetros:

cTitulo	Título do relatório
cCabec1	String contendo as informações da primeira linha do cabeçalho
cCabec2	String contendo as informações da segunda linha do cabeçalho
cNomeProg	Nome do programa de impressão do relatório.
nTamanho	Tamanho do relatório em colunas (80, 132 ou 220)
nCompress	Indica se impressão será comprimida (15) ou normal (18).
aCustomText	Texto específico para o cabeçalho, substituindo a estrutura padrão do sistema.
lPerg	Permite a supressão da impressão das perguntas do relatório, mesmo que o parâmetro MV_IMPSX1 esteja definido como "S"
cLogo	Redefine o bitmap que será impresso no relatório, não necessitando que ele esteja no formato padrão da Microsiga: "LGRL"+SM0->M0_CODIGO+SM0->M0_CODFIL+".BMP"

2.1.5. RODA()

A função RODA() imprime o rodapé da página do relatório, o que pode ser feito a cada página, ou somente ao final da impressão.

- Sintaxe: Roda(uPar01, uPar02, cSize)
- Parâmetros:

uPar01	Não é mais utilizado
uPar02	Não é mais utilizado
cSize	Tamanho do relatório ("P","M","G")

2.1.6. Pergunte()

A função PERGUNTE() inicializa as variáveis de pergunta (mv_par01,...) baseada na pergunta cadastrado no Dicionário de Dados (SX1). Se o parâmetro lAsk não for especificado ou for verdadeiro será exibida a tela para edição da pergunta e se o usuário confirmar as variáveis serão atualizadas e a pergunta no SX1 também será atualizada.

- Sintaxe: Pergunte(cPergunta , [lAsk] , [cTitle])
- Parâmetros:

cPergunta	Pergunta cadastrada no dicionário de dados (SX1) a ser utilizada.
Ask	Indica se exibirá a tela para edição.
cTitle	Título do diálogo.

- Retorno:

Lógico	Indica se a tela de visualização das perguntas foi confirmada (.T.) ou cancelada (.F.)
---------------	--

2.1.7. AjustaSX1()

A função AJUSTASX1() permite a inclusão simultânea de vários itens de perguntas para um grupo de perguntas no SX1 da empresa ativa.

- Sintaxe: AJUSTASX1(cPerg, aPergs)
- Parâmetros:

cPerg	Grupo de perguntas do SX1 (X1_GRUPO)
aPergs	Array contendo a estrutura dos campos que serão gravados no SX1.

- Estrutura – Item do array aPerg:

Posição	Campo	Tipo	Descrição
01	X1_PERGUNT	Caractere	Descrição da pergunta em português
02	X1_PERSPA	Caractere	Descrição da pergunta em espanhol
03	X1_PERENG	Caractere	Descrição da pergunta em inglês
04	X1_VARIAVL	Caractere	Nome da variável de controle auxiliar (mv_ch)
05	X1_TIPO	Caractere	Tipo do parâmetro
06	X1_TAMANHO	Numérico	Tamanho do conteúdo do parâmetro
07	X1_DECIMAL	Numérico	Número de decimais para conteúdos numéricos
08	X1_PRESEL	Numérico	Define qual opção do combo é a padrão para o parâmetro.

09	X1_GSC	Caractere	Define se a pergunta será do tipo G – Get ou C – Choice (combo)
10	X1_VALID	Caractere	Expressão de validação do parâmetro
11	X1_VAR01	Caractere	Nome da variável MV_PAR+”Ordem” do parâmetro
12	X1_DEF01	Caractere	Descrição da opção 1 do combo em português
13	X1_DEFSPA1	Caractere	Descrição da opção 1 do combo em espanhol
14	X1_DEFENG1	Caractere	Descrição da opção 1 do combo em inglês
15	X1_CNT01	Caractere	Conteúdo padrão ou ultimo conteúdo definido como respostas para a pergunta.
16	X1_VAR02	Caractere	Não é informado
17	X1_DEF02	Caractere	Descrição da opção X do combo em português
18	X1_DEFSPA2	Caractere	Descrição da opção X do combo em espanhol
19	X1_DEFENG2	Caractere	Descrição da opção X do combo em inglês
20	X1_CNT02	Caractere	Não é informado
21	X1_VAR03	Caractere	Não é informado
22	X1_DEF03	Caractere	Descrição da opção X do combo em português
23	X1_DEFSPA3	Caractere	Descrição da opção X do combo em espanhol
24	X1_DEFENG3	Caractere	Descrição da opção X do combo em inglês
25	X1_CNT03	Caractere	Não é informado
26	X1_VAR04	Caractere	Não é informado
27	X1_DEF04	Caractere	Descrição da opção X do combo em português
28	X1_DEFSPA4	Caractere	Descrição da opção X do combo em espanhol
29	X1_DEFENG4	Caractere	Descrição da opção X do combo em inglês
30	X1_CNT04	Caractere	Não é informado
31	X1_VAR05	Caractere	Não é informado
32	X1_DEF05	Caractere	Descrição da opção X do combo em português
33	X1_DEFSPA5	Caractere	Descrição da opção X do combo em espanhol
34	X1_DEFENG5	Caractere	Descrição da opção X do combo em inglês
35	X1_CNT05	Caractere	Não é informado
36	X1_F3	Caractere	Código da consulta F3 vinculada ao parâmetro
37	X1_GRPSXG	Caractere	Código do grupo de campos SXG para atualização automática, quando o grupo for alterado.
38	X1_PYME	Caractere	Se a pergunta estará disponível no ambiente Pyme
39	X1_HELP	Caractere	Conteúdo do campo X1_HELP
40	X1_PICTURE	Caractere	Picture de formatação do conteúdo do campo.
41	aHelpPor	Array	Vetor simples contendo as linhas de help em português para o parâmetro. Trabalhar com linhas de até 40 caracteres.
42	aHelpEng	Array	Vetor simples contendo as linhas de help em inglês para o parâmetro. Trabalhar com linhas de até 40 caracteres.
43	aHelpSpa	Array	Vetor simples contendo as linhas de help em espanhol para o parâmetro. Trabalhar com linhas de até 40 caracteres.

2.1.8. PutSX1()

A função PUTSX1() permite a inclusão de um único item de pergunta em um grupo de definido no Dicionário de Dados (SX1). Todos os vetores contendo os textos explicativos da pergunta devem conter até 40 caracteres por linha.

- Sintaxe: PutSX1(cGrupo, cOrdem, cPergunt, cPerSpa, cPerEng, cVar, cTipo, nTamanho, nDecimal, nPresel, cGSC, cValid, cF3, cGrpSxg, cPyme, cVar01, cDef01, cDefSpa1, cDefEng1, cCnt01, cDef02, cDefSpa2, cDefEng2, cDef03, cDefSpa3, cDefEng3, cDef04, cDefSpa4, cDefEng4, cDef05, cDefSpa5, cDefEng5, aHelpPor, aHelpEng, aHelpSpa, cHelp)
- Parâmetros:

cGrupo	Grupo de perguntas do SX1 (X1_GRUPO)
cOrdem	Ordem do parâmetro no grupo (X1_ORDEM)
cPergunt	Descrição da pergunta em português
cPerSpa	Descrição da pergunta em espanhol
cPerEng	Descrição da pergunta em inglês
cVar	Nome da variável de controle auxiliar (X1_VARIAVL)
cTipo	Tipo do parâmetro
nTamanho	Tamanho do conteúdo do parâmetro
nDecimal	Número de decimais para conteúdos numéricos
nPresel	Define qual opção do combo é a padrão para o parâmetro.
cGSC	Define se a pergunta será do tipo G – Get ou C – Choice (combo)
cValid	Expressão de validação do parâmetro
cF3	Código da consulta F3 vinculada ao parâmetro
cGrpSxg	Código do grupo de campos SXG para atualização automática, quando o grupo for alterado.
cPyme	Se a pergunta estará disponível no ambiente Pyme
cVar01	Nome da variável MV_PAR+”Ordem” do parâmetro.
cDef01	Descrição da opção 1 do combo em português
cDefSpa1	Descrição da opção 1 do combo em espanhol
cDefEng1	Descrição da opção 1 do combo em inglês
cCnt01	Conteúdo padrão ou ultimo conteúdo definido como respostas para este item
cDef0x	Descrição da opção X do combo em português
cDefSpax	Descrição da opção X do combo em espanhol
cDefEngx	Descrição da opção X do combo em inglês
aHelpPor	Vetor simples contendo as linhas de help em português para o parâmetro.
aHelpEng	Vetor simples contendo as linhas de help em inglês para o parâmetro.

aHelpSpa	Vetor simples contendo as linhas de help em espanhol para o parâmetro.
cHelp	Conteúdo do campo X1_HELP

2.2. Estrutura de Relatórios Baseados na SetPrint()

Neste tópico será demonstrada a construção de relatório não gráfico baseado no uso da função SetPrint() o qual atende os formatos de base de dados ISAM e Topconnect, porém não contemplando a tecnologia Protheus Embedded SQL.

Estrutura do programa

Linhas	Programa
1	Função principal;
2	Declaração e atribuição de variáveis;
3	Atualização do arquivo de perguntas através da função específica CriaSX1();
4	Definição as perguntas através da função Pergunte();
5	Definição das ordens disponíveis para impressão do relatório;
6	Chamada da função SetPrint;
7	Atualização das configurações de impressão com a função SetDefault();
8	Execução da rotina de impressão através da função RptStatus()
9	Fim da função principal.
10	Função de processamento e impressão do relatório
11	Declaração e atribuição de variáveis;
12	Definição dos filtros de impressão, avaliando o bando de dados em uso pela aplicação;
13	Atualização da régua de processamento com a quantidade de registros que será processada;
14	Estrutura principal de repetição para impressão dos dados do relatório;
15	Controle da impressão do cabeçalho utilizando a função Cabec();
16	Impressão dos totais do relatório;
17	Impressão do rodapé da última página do relatório utilizando a função Roda();
18	Limpeza dos arquivos e índices temporários criados para o processamento();
19	Tratamento da visualização do relatório (impressão em disco) através da função OurSpool()
20	Tratamentos adicionais ao relatório, de acordo com necessidades específicas;
21	Liberação do buffer de impressão, seja para impressora, seja para limpeza do conteúdo visualizado em tela, utilizando a função MS_FLUSH()

- 22 **Fim da função de processamento e impressão do relatório**
- 23 **Função de atualização do arquivo de perguntas**
- 24 Declaração e atribuição de variáveis;
- 25 Opção 01: Adição individual de cada pergunta no SX1 utilizando a função PUTSX1()
Criação de um array individual no formato utilizado pela PUTSX1() contendo apenas as informações da pergunta que será adicionada no SX1.
- 25 Opção 02: Adição de um grupo de perguntas no SX1 utilizando a função AJUSTASX1()
Criação de um array no formato utilizado pela AJUSTASX1() contendo todas as perguntas que serão atualizadas.
- 26 **Fim da função de atualização do arquivo de perguntas**

Função Principal

```
//+-----+
//| Rotina | Inform | Autor | Robson Luiz (rleg) | Data | 01.01.07 |
//+-----+
//| Descr. | Rotina para gerar relatório utilizando as funções |
//|      | SetPrint() e SetDefault(). |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
```

User Function INFORM()

```
//+-----+
//| Declarações de variáveis
//+-----+
```

```
Local cDesc1 := "Este relatório irá imprimir informações do contas a pagar conforme"
Local cDesc2 := "parâmetros informado. Será gerado um arquivo no diretório "
Local cDesc3 := "Spool - INFORM_????XLS, onde ???? e o nome do usuário."
```

```
Private cString := "SE2"
Private Tamanho := "M"
```

```

Private aReturn := { "Zebrado",2,"Administração",2,2,1,"",1 }
Private wnrel   := "INFORM"
Private NomeProg := "INFORM"
Private nLastKey := 0
Private Limite   := 132
Private Titulo   := "Título a Pagar - Ordem de "
Private cPerg    := "INFORM"
Private nTipo    := 0
Private cbCont   := 0
Private cbTxt    := "registro(s) lido(s)"
Private Li       := 80
Private m_pag    := 1
Private aOrd     := {}
Private Cabec1   := "PREFIXO TITULO PARCELA TIP EMISSAO VENCTO  VENCTO"
Private Cabec1 += "REAL VLR. ORIGINAL    PAGO          SALDO "
Private Cabec2   := ""
/*
+-----
| Parâmetros do aReturn
+-----
aReturn - Preenchido pelo SetPrint()
aReturn[1] - Reservado para formulário
aReturn[2] - Reservado para numero de vias
aReturn[3] - Destinatário
aReturn[4] - Formato 1=Paisagem 2=Retrato
aReturn[5] - Mídia 1-Disco 2=Impressora
aReturn[6] - Porta ou arquivo 1-Lpt1... 4-Com1...
aReturn[7] - Expressão do filtro
aReturn[8] - Ordem a ser selecionada
aReturn[9] [10] [n] - Campos a processar se houver
*/

AADD( aOrd, "Fornecedor" )
AADD( aOrd, "Titulo" )
AADD( aOrd, "Emissão" )
AADD( aOrd, "Vencimento" )
AADD( aOrd, "Vencto. Real" )

```

```
//Parâmetros de perguntas para o relatório
//+-----+
//| mv_par01 - Fornecedor de   ? 999999          |
//| mv_par02 - Fornecedor ate   ? 999999          |
//| mv_par03 - Tipo de         ? XXX              |
//| mv_par04 - Tipo ate        ? XXX              |
//| mv_par05 - Vencimento de    ? 99/99/99        |
//| mv_par06 - Vencimento ate   ? 99/99/99        |
//| mv_par07 - Aglut.Fornecedor ? Sim/Não         |
//+-----+
CriaSx1()

//+-----+
//| Disponibiliza para usuário digitar os parâmetros
//+-----+
Pergunte(cPerg,.F.)
//cPerg -> Nome do grupo de perguntas, .T. mostra a tela;;
// .F. somente carrega as variáveis

//+-----+
//| Solicita ao usuário a parametrização do relatório.
//+-----+
wnrel := SetPrint(cString,wnrel,cPerg,@Titulo,cDesc1,cDesc2,cDesc3,.F.,aOrd,.F. ;
Tamanho,.F.,.F.)
//SetPrint(cAlias,cNome,cPerg,cDesc,cCnt1,cCnt2,cCnt3,lDic,aOrd,lCompres;;
//cSize,aFilter,lFiltro,lCrystal,cNameDrv,lNoAsk,lServer,cPortToPrint)

//+-----+
//| Se teclar ESC, sair
//+-----+
If nLastKey == 27
Return
Endif

//+-----+
//| Estabelece os padrões para impressão, conforme escolha do usuário
//+-----+
```

```
SetDefault(aReturn,cString)
```

```
//+-----
//| Verificar se será reduzido ou normal
//+-----
nTipo := IIF(aReturn[4] == 1, 15, 18)
```

```
//+-----
//| Se teclar ESC, sair
//+-----
If nLastKey == 27
Return
Endif
```

```
//+-----
//| Chama função que processa os dados
//+-----
RptStatus({|IEnd| ImpRel(@IEnd) }, Titulo, "Processando e imprimindo dados,,
aguarde...", .T. )
```

```
Return
```

Função de processamento e impressão

```
//+-----+
//| Rotina | ImpRel | Autor | Robson Luiz (rleg) | Data | 01.01.07 |
//+-----+
//| Descr. | Rotina de processamento e impressão. |
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+
```

```
Static Function ImpRel(IEnd)
```

```
Local nIndice := 0
Local cArq := ""
Local cIndice := ""
```

```
Local cFiltro := ""
Local aCol := {}
Local cFornec := ""
Local nValor := 0
Local nPago := 0
Local nSaldo := 0
Local nT_Valor := 0
Local nT_Pago := 0
Local nT_Saldo := 0
Local cArqExcel := ""
Local cAliasImp
Local oExcelApp

Titulo += aOrd[aReturn[8]]

#IFDEF TOP
cAliasImp := "SE2"

cFiltro := "E2_FILIAL == '"+xFilial("SE2")+"' "
cFiltro += ".And. E2_FORNECE >= '"+mv_par01+"' "
cFiltro += ".And. E2_FORNECE <= '"+mv_par02+"' "
cFiltro += ".And. E2_TIPO >= '"+mv_par03+"' "
cFiltro += ".And. E2_TIPO <= '"+mv_par04+"' "
cFiltro += ".And. Dtos(E2_VENCTO) >= '"+Dtos(mv_par05)+"' "
cFiltro += ".And. Dtos(E2_VENCTO) <= '"+Dtos(mv_par06)+"' "

If aReturn[8] == 1 //Fornecedor
    cIndice := "E2_FORNECE+E2_LOJA+E2_NUM"
Elseif aReturn[8] == 2 //Titulo
    cIndice := "E2_NUM+E2_FORNECE+E2_LOJA"
Elseif aReturn[8] == 3 //Emissao
    cIndice := "Dtos(E2_EMISSAO)+E2_FORNECE+E2_LOJA"
Elseif aReturn[8] == 4 //Vencimento
    cIndice := "Dtos(E2_VENCTO)+E2_FORNECE+E2_LOJA"
Elseif aReturn[8] == 5 //Vencimento Real
    cIndice := "Dtos(E2_VENCREA)+E2_FORNECE+E2_LOJA"
Endif
```

```
cArq := CriaTrab(NIL,.F.)
dbSelectArea(cAliasImp)
IndRegua(cAliasImp,cArq,cIndice,,cFiltro)
nIndice := RetIndex()
nIndice := nIndice + 1
dbSetIndex(cArq+OrdBagExt())
dbSetOrder(nIndice)
#ELSE
cAliasImp := GetNextAlias()

cQuery := "SELECT "
cQuery += "E2_PREFIXO, E2_NUM, E2_PARCELA, E2_TIPO, E2_FORNECE, E2_LOJA, "
cQuery += "E2_NOMFOR, "
cQuery += "E2_EMISSAO, E2_VENCTO, E2_VENCREA, E2_VALOR, E2_SALDO "
cQuery += "FROM "+RetSqlName("SE2")+ " "
cQuery += "WHERE E2_FILIAL = '"+xFilial("SE2")+"' "
cQuery += "AND E2_FORNECE >= '"+mv_par01+"' "
cQuery += "AND E2_FORNECE <= '"+mv_par02+"' "
cQuery += "AND E2_TIPO >= '"+mv_par03+"' "
cQuery += "AND E2_TIPO <= '"+mv_par04+"' "
cQuery += "AND E2_VENCTO >= '"+Dtos(mv_par05)+"' "
cQuery += "AND E2_VENCTO <= '"+Dtos(mv_par06)+"' "
cQuery += "AND D_E_L_E_T_ <> '*' "
cQuery += "ORDER BY "

If aReturn[8] == 1 //Fornecedor
    cQuery += "E2_FORNECE,E2_LOJA,E2_NUM"
Elseif aReturn[8] == 2 //Titulo
    cQuery += "E2_NUM,E2_FORNECE,E2_LOJA"
Elseif aReturn[8] == 3 //Emissao
    cQuery += "E2_EMISSAO,E2_FORNECE,E2_LOJA"
Elseif aReturn[8] == 4 //Vencimento
    cQuery += "E2_VENCTO,E2_FORNECE,E2_LOJA"
Elseif aReturn[8] == 5 //Vencimento Real
    cQuery += "E2_VENCREA,E2_FORNECE,E2_LOJA"
Endif
```

```
dbUseArea( .T., "TOPCONN", TcGenQry(,,cQuery), cAliasImp, .T., .F. )
```

```
dbSelectArea(cAliasImp)
```

```
/* Instrução SQL Embedded
```

```
-----
```

```
If aReturn[8] == 1 //Fornecedor
```

```
    cOrder := "E2_FORNECE,E2_LOJA,E2_NUM"
```

```
Elseif aReturn[8] == 2 //Titulo
```

```
    cOrder := "E2_NUM,E2_FORNECE,E2_LOJA"
```

```
Elseif aReturn[8] == 3 //Emissao
```

```
    cOrder := "E2_EMISSAO,E2_FORNECE,E2_LOJA"
```

```
Elseif aReturn[8] == 4 //Vencimento
```

```
    cOrder := "E2_VENCTO,E2_FORNECE,E2_LOJA"
```

```
Elseif aReturn[8] == 5 //Vencimento Real
```

```
    cOrder := "E2_VENCREA,E2_FORNECE,E2_LOJA"
```

```
Endif
```

```
BeginSQL Alias cAliasImp
```

```
    Column E2_EMISSAO           As Date
```

```
    Column E2_VENCTO           As Date
```

```
    Column E2_VENCREA          As Date
```

```
    Column E2_VALOR            As Numeric(12)
```

```
    Column E2_SALDO            As Numeric(12)
```

```
    %NoParser%
```

```
        SELECT                                E2_PREFIXO, E2_NUM, E2_PARCELA, E2_TIPO,
E2_FORNECE,
```

```
        E2_LOJA, E2_NOMFOR, E2_EMISSAO, E2_VENCTO, E2_VENCREA, E2_VALOR,
        E2_SALDO
```

```
        FROM                                %Table:SE2
```

```
        WHERE
```

```
        E2_FILIAL = %xFilial% AND
```

```
        E2_FORNECE BETWEEN %Exp:mv_par01% AND %Exp:mv_par02% AND
```

```
        E2_TIPO BETWEEN%Exp:mv_par03% AND %Exp:mv_par04% AND
```

```
        E2_VENCTO BETWEEN %Exp:mv_par05% AND %Exp:mv_par06% AND
```

```
        %NotDel%
```

```
        ORDER BY %Order:cOrder%
```

```
EndSQL
*/
#ENDIF

dbGoTop()
SetRegua(0)

//+-----
//| Coluna de impressão
//+-----
AADD( aCol, 004 ) //Prefixo
AADD( aCol, 012 ) //Titulo
AADD( aCol, 024 ) //Parcela
AADD( aCol, 031 ) //Tipo
AADD( aCol, 036 ) //Emissao
AADD( aCol, 046 ) //Vencimento
AADD( aCol, 058 ) //Vencimento Real
AADD( aCol, 070 ) //Valor Original
AADD( aCol, 090 ) //Pago
AADD( aCol, 110 ) //Saldo

cFornec := (cAliasImp)->E2_FORNECE+(cAliasImp)->E2_LOJA

While !Eof() .And. !End

If Li > 55
    Cabec(Titulo,Cabec1,Cabec2,NomeProg,Tamanho,nTipo)
Endif

@ Li, aCol[1] PSay "Cod/Loj/Nome: "+(cAliasImp)->E2_FORNECE+;
"- "+(cAliasImp)->E2_LOJA+" "+(cAliasImp)->E2_NOMFOR
Li ++

While !Eof() .And. !End .And.;
(cAliasImp)->E2_FORNECE+(cAliasImp)->E2_LOJA == cFornec

IncRegua()
```



```

If Li > 55
Cabec(Titulo,Cabec1,Cabec2,NomeProg,Tamanho,nTipo)
Endif

If mv_par07 == 2
@ Li, aCol[1] PSay (cAliasImp)->E2_PREFIXO
                                @ Li, aCol[2] PSay (cAliasImp)->E2_NUM
                                @ Li, aCol[3] PSay (cAliasImp)->E2_PARCELA
                                @ Li, aCol[4] PSay (cAliasImp)->E2_TIPO
                                @ Li, aCol[5] PSay (cAliasImp)->E2_EMISSAO
                                @ Li, aCol[6] PSay (cAliasImp)->E2_VENCTO
                                @ Li, aCol[7] PSay (cAliasImp)->E2_VENCREA
                                @ Li, aCol[8] PSay (cAliasImp)->E2_VALOR ;

                                PICTURE "@E 99,999,999,999.99"

                                @ Li, aCol[9] PSay (cAliasImp)->E2_VALOR -;

                                (cAliasImp)->E2_SALDO ;
                                PICTURE "@E 99,999,999,999.99"

                                @ Li, aCol[10] PSay (cAliasImp)->E2_SALDO ;

                                PICTURE "@E 99,999,999,999.99"

                                Li ++

                                Endif

                                nValor += (cAliasImp)->E2_VALOR
                                nPago += ((cAliasImp)->E2_VALOR-(cAliasImp)->E2_SALDO)
                                nSaldo += (cAliasImp)->E2_SALDO

                                nT_Valor += (cAliasImp)->E2_VALOR
                                nT_Pago += ((cAliasImp)->E2_VALOR-(cAliasImp)->E2_SALDO)
                                nT_Saldo += (cAliasImp)->E2_SALDO

                                dbSkip()

                                End

                                @ Li, 000 PSay Replicate("-",Limite)
                                Li ++
                                @ Li, aCol[1] PSay "TOTAL....."
                                @ Li, aCol[8] PSay nValor PICTURE "@E 99,999,999,999.99"

```

```
@ Li, aCol[9] PSay nPago PICTURE "@E 99,999,999,999.99"
```

```
@ Li, aCol[10] PSay nSaldo PICTURE "@E 99,999,999,999.99"
```

```
Li +=2
```

```
cFornec := (cAliasImp)->E2_FORNECE+(cAliasImp)->E2_LOJA
```

```
nValor := 0
```

```
nPago := 0
```

```
nSaldo := 0
```

```
End
```

```
If !End
```

```
@ Li, aCol[1] PSay cCancel
```

```
Return
```

```
Endif
```

```
@ Li, 000 PSay Replicate("=",Limite)
```

```
Li ++
```

```
@ Li, aCol[1] PSay "TOTAL GERAL....."
```

```
@ Li, aCol[8] PSay nT_Valor PICTURE "@E 99,999,999,999.99"
```

```
@ Li, aCol[9] PSay nT_Pago PICTURE "@E 99,999,999,999.99"
```

```
@ Li, aCol[10] PSay nT_Saldo PICTURE "@E 99,999,999,999.99"
```

```
If Li <> 80
```

```
Roda(cbCont,cbTxt,Tamanho)
```

```
Endif
```

```
//+-----
```

```
//| Gera arquivo do tipo .DBF com extensão .XLS p/ usuário abrir no Excel
```

```
//+-----
```

```
cArqExcel := __RELDIR+NomeProg+"_"+Substr(cUsuario,7,4)+".XLS"
```

```
Copy To &cArqExcel
```

```
#IFDEF TOP
```

```
dbSelectArea(cAliasImp)
```

```
RetIndex(cAliasImp)
```

```
Set Filter To
```

```
#ELSE
dbSelectArea(cAliasImp)
dbCloseArea()
#ENDIF
dbSetOrder(1)
dbGoTop()

If aReturn[5] == 1
Set Printer TO
dbCommitAll()
OurSpool(wnrel)
EndIf

//+-----
//| Abrir planilha MS-Excel
//+-----
If mv_par08 == 1
__CopyFile(cArqExcel,"c:\"+NomeProg+"_"+Substr(cUsuario,7,4)+".XLS")
If ! ApOleClient("MsExcel")
MsgAlert("MsExcel não instalado")
Return
Endif
oExcelApp := MsExcel():New()
oExcelApp:WorkBooks:Open( "c:\"+NomeProg+"_"+Substr(cUsuario,7,4)+".XLS" )
oExcelApp:SetVisible(.T.)
Endif

Ms_Flush()

Return
```

Função para gerar o grupo de parâmetros no SX1

```
//+-----+
//| Rotina | CriaSX1 | Autor | Robson Luiz (rleg)| Data | 01.01.07 |
//+-----+
//| Descr. | Rotina para criar o grupo de parâmetros. |
```

```
//+-----+
//| Uso   | Para treinamento e capacitação. |
//+-----+

Static Function CriaSx1()
Local aP := {}
Local i := 0
Local cSeq
Local cMvCh
Local cMvPar
Local aHelp := {}

/*****
Parâmetros da função padrão
-----

PutSX1(cGrupo,,cOrdem,,
cPergunt,cPerSpa,cPerEng,,
cVar,,
cTipo,,
nTamanho,,
nDecimal,,
nPresel,,
cGSC,,
cValid,,
cF3,,
cGrpSxg,,
cPyme,,
cVar01,,
cDef01,cDefSpa1,cDefEng1,,
cCnt01,,
cDef02,cDefSpa2,cDefEng2,,
cDef03,cDefSpa3,cDefEng3,,
cDef04,cDefSpa4,cDefEng4,,
cDef05,cDefSpa5,cDefEng5,,
aHelpPor,aHelpEng,aHelpSpa,,
cHelp)
```

Característica do vetor p/ utilização da função SX1

```

-----
[n,1] --> texto da pergunta
[n,2] --> tipo do dado
[n,3] --> tamanho
[n,4] --> decimal
[n,5] --> objeto G=get ou C=choice
[n,6] --> validação
[n,7] --> F3
[n,8] --> definição 1
[n,9] --> definição 2
[n,10] --> definição 3
[n,11] --> definição 4
[n,12] --> definição 5
*** /
AADD(aP,{"Fornecedor de","C",6,0,"G","", "SA2", "" , "" , "" , "" , ""})
AADD(aP,{"Fornecedor ate","C",6,0,"G","(mv_par02>=mv_par01)","SA2",;
"" , "" , "" , "" , ""})
AADD(aP,{"Tipo de","C",3,0,"G","", "05", "" , "" , "" , "" , ""})
AADD(aP,{"Tipo ate","C",3,0,"G","(mv_par04>=mv_par03)","05", "" ,;
"" , "" , "" , "" , ""})
AADD(aP,{"Vencimento de","D",8,0,"G","", "" , "" , "" , "" , ""})
AADD(aP,{"Vencimento ate","D",8,0,"G","(mv_par06>=mv_par05)","" ,;
"" , "" , "" , "" , ""})
AADD(aP,{"Aglutinar pagto.de fornec.","N",1,0,"C","", "" ,;
"Sim","Não", "" , "" , ""})
AADD(aP,{"Abrir planilha MS-Excel" , "N",1,0,"C","", "" ,;
"Sim","Não", "" , "" , ""})
AADD(aHelp,{"Informe o código do fornecedor.", "inicial."})
AADD(aHelp,{"Informe o código do fornecedor.", "final."})
AADD(aHelp,{"Tipo de título inicial."})
AADD(aHelp,{"Tipo de título final."})
AADD(aHelp,{"Digite a data do vencimento inicial."})
AADD(aHelp,{"Digite a data do vencimento final."})
AADD(aHelp,{"Aglutinar os títulos do mesmo forne-";
"cedor totalizando seus valores."})
AADD(aHelp,{"Será gerada uma planilha para ";
"MS-Excel, abrir esta planilha?"})

```

```
For i:=1 To Len(aP)
  cSeq := StrZero(i,2,0)
  cMvPar := "mv_par"+cSeq
  cMvCh := "mv_ch"+IIF(i<=9,Chr(i+48),Chr(i+87))
```

```
PutSx1(cPerg,;
  cSeq,;
  aP[i,1],aP[i,1],aP[i,1],;
  cMvCh,;
  aP[i,2],;
  aP[i,3],;
  aP[i,4],;
  0,;
  aP[i,5],;
  aP[i,6],;
  aP[i,7],;
  "",;
  "",;
  cMvPar,;
  aP[i,8],aP[i,8],aP[i,8],;
  "",;
  aP[i,9],aP[i,9],aP[i,9],;
  aP[i,10],aP[i,10],aP[i,10],;
  aP[i,11],aP[i,11],aP[i,11],;
  aP[i,12],aP[i,12],aP[i,12],;
  aHelp[i],;
  {},,;
  {},,;
  "")
Next i
```

```
Return
```

Exercício

Implementar um relatório que forneça uma listagem de uma nota fiscal de entrada e seus itens.

2.3. Introdução à relatórios gráficos

2.3.1. Introdução

Finalidade

O Protheus oferece o recurso personalização para alguns relatórios de cadastros e movimentações do sistema. Ele tem como principais funcionalidades a definição de cores, estilos, tamanho, fontes, quebras, máscara das células para cada seção, criação de fórmulas e funções (Soma, Média, etc.), possibilidade de salvar as configurações por usuário e criação de gráficos.

Com a funcionalidade de Relatórios Personalizáveis, o usuário pode modificar os relatórios padrões, criando seu próprio layout.

Vale lembrar que nem todos os relatórios são personalizáveis. Por exemplo, relatórios que tenham layout pré-definidos por lei e formulários (boletos, notas-fiscais, etc) não poderão ser alterados.

Os relatórios personalizados são gravados com extensão .PRT, diferenciando-se dos relatórios padrões que recebem a extensão .###R.

Descrição

O TReport é uma classe de impressão que substitui as funções SetPrint, SetDefault, RptStatus e Cabec.

A classe TReport permite que o usuário personalize as informações que serão apresentadas no relatório, alterando fonte (tipo, tamanho, etc), cor, tipo de linhas, cabeçalho, rodapé, etc.

Estrutura do componente TReport:

- O relatório (TReport) contém 1 ou mais seções (TRSection);
- Uma seção (TRSection) pode conter 1 ou mais seções;
- A seção (TRSection) contém células pré-definidas e células selecionadas pelo usuário;
- A seção (TRSection) também contém as quebras (TRBreak) para impressão de totalizadores (TRFunction);
- Os totalizadores são incluídos pela seção que automaticamente inclui no relatório (TReport).

Pré-Requisitos

Para utilizar o TReport, verifique se o seu repositório está com o Release 4 do Protheus-8, ou versão superior.

A função TRepInUse() verifica se a lib do TReport está liberada no repositório em uso. O retorno é uma variável lógica.

```
#include "protheus.ch"  
User Function MyReport()  
Local oReport
```

```

If TRepInUse()                                     //verifica se a opção relatórios personalizáveis
está disponível

    Pergunte("MTR025",.F.)

    oReport := ReportDef()
    oReport:PrintDialog()

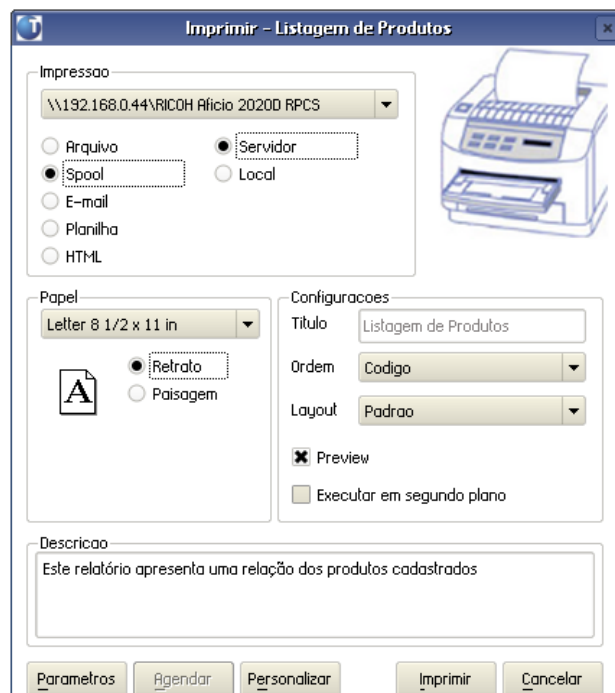
EndIf
Return
    
```

Verifique também o parâmetro MV_TReport. Para utilizar os relatórios personalizáveis, o parâmetro MV_TREPORT (tipo numérico) deve ser alterado no ambiente Configurador, conforme uma das opções que seguem:

- 1 = utiliza relatório no formato tradicional (antigo);
- 2 = utiliza relatório personalizável;
- 3 = pergunta qual relatório será utilizado: tradicional (antigo) ou personalizável.

2.3.2. Impressão do relatório personalizável

Cada componente da tela de impressão do TReport, deve ser configurado no programa, para que o usuário tenha acesso às personalizações:



2.3.2.1. Parâmetros de impressão

A caixa de listagem apresentada deve ser utilizada conforme o meio de saída do relatório. Veja a seguir.

Impressão

Arquivo

O relatório será gravado em disco com o nome apresentado. Caso seja escolhida a opção "Servidor" ele será gravado no diretório determinado na senha do usuário, através do configurador, sendo este sempre no servidor (padrão \SPOOL\). Na escolha da opção "Local" será aberta uma janela para que seja escolhido o local onde o relatório será gravado na máquina do usuário.

O relatório gerado a partir desta opção pode ser impresso ou enviado por e-mail após ser apresentado na tela.

Spool

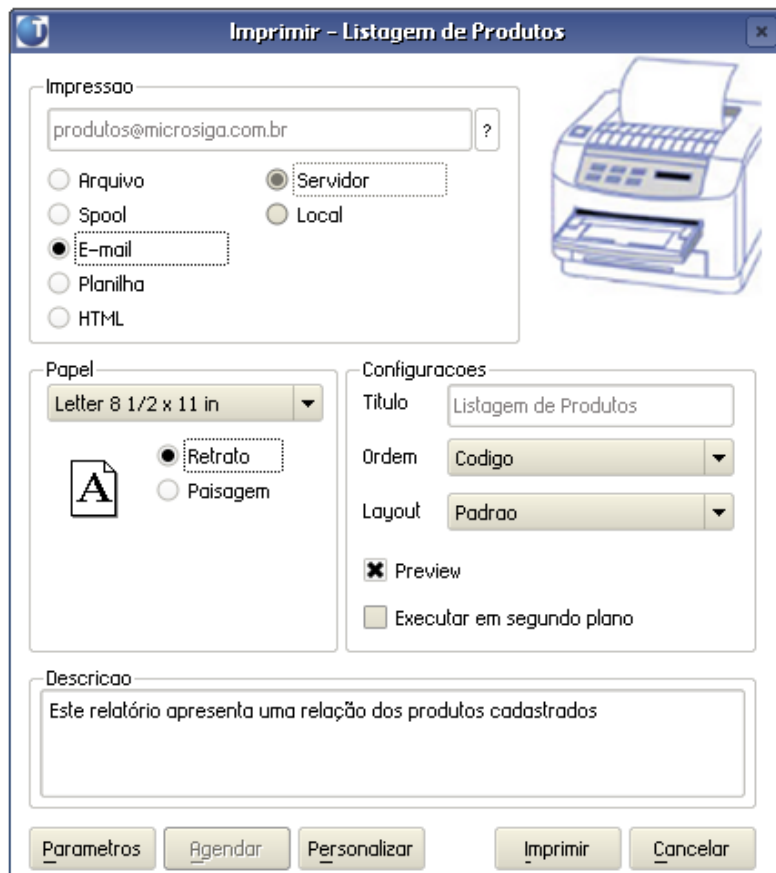
Direciona o relatório para impressão via configuração do Windows® das impressoras instaladas.

E-mail

Envia o relatório por e-mail (Internet). Para isto, devem ser configurados os seguintes parâmetros no Ambiente Configurador:

- MV_RELACNT
Define a conta de e-mail para identificar a proveniência dos relatórios.
Exemplo: relprotheus@microsiga.com.br
- MV_RELPSW
Define a senha da conta de e-mail para envio dos relatórios.
- MV_RELSERV
Define o servidor da conta de e-mail para o envio do relatório.
Exemplo: smtp.microsiga.com.br

Quando selecionada esta opção, deve-se informar, no campo em destaque na figura abaixo, o e-mail para o qual o relatório deve ser remetido.



O Protheus Server pode também ser executado como um servidor Web, respondendo a requisições HTTP. No momento destas requisições, pode executar rotinas escritas em ADVPL como processos individuais, enviando o resultado das funções como retorno das requisições para o cliente HTTP (como por exemplo, um Browser de Internet). Qualquer rotina escrita em ADVPL que não contenha comandos de interface pode ser executada através de requisições HTTP. O Protheus permite a compilação de arquivos HTML contendo código ADVPL embutido. São os chamados arquivos ADVPL ASP, para a criação de páginas dinâmicas.

- **Programação TelNet**

TelNet é parte da gama de protocolos TCP/IP que permite a conexão a um computador remoto através de uma aplicação cliente deste protocolo. O PROTHEUS Server pode emular um terminal TelNet, através da execução de rotinas escritas em ADVPL. Ou seja, pode-se escrever rotinas ADVPL cuja interface final será um terminal TelNet ou um coletor de dados móvel.

Papel

Tamanho do papel

Selecione o tamanho do papel em que o relatório será impresso.

As especificações de tamanho do papel são as do padrão do mercado, conforme o formato escolhido, o Protheus irá ajustar a impressão.

Formato da impressão

Selecione o formato de impressão, clicando nos botões de opção “Retrato” ou “Paisagem”, fazendo assim que o relatório seja impresso na orientação vertical ou horizontal, respectivamente.

Configurações

Título

Caso queira alterar a opção sugerida pelo sistema, digite o cabeçalho do relatório.

Ordem

Escolha a ordem em que as informações serão apresentadas no relatório, clicando em uma das chaves disponíveis.

Layout

Permite selecionar o modelo de relatório para impressão, à medida que novos leiautes forem gravados para um relatório, seus nomes serão listados nessa caixa.

Preview

Faz a exibição do relatório gerado na tela, possibilitando, na sequência, o seu envio para impressora ou a cancelamento da impressão.

Executar em segundo plano

Essa opção permite que o relatório seja gerado e enviado para a fila de impressão, enquanto o usuário pode executar outras tarefas no sistema.

2.3.3. Personalização

É possível configurar-se as colunas do lay-out do relatório, bem como os acumuladores, cabeçalhos e linhas.

Estão disponíveis para personalização também a fonte, tamanho, cores, e etc.


Imprimir - Relação de Clientes

Impressão

\\printserver\TecMono Lexmark T632 (PCL) ▼

☐ Arquivo
 ☒ Servidor
 ☒ Spool
 ☐ Local
 ☐ Porta
 ☐ E-mail

Formato


☒ Retrato
 ☐ Paisagem

Configurações

Título: Relação de Clientes

Cópias: 1

☒ Preview

Descrição

Este relatório irá imprimir a Relação de Clientes de acordo com os parâmetros solicitados pelo usuário. Para maiores informações sobre este relatório consulte o Help (F1)

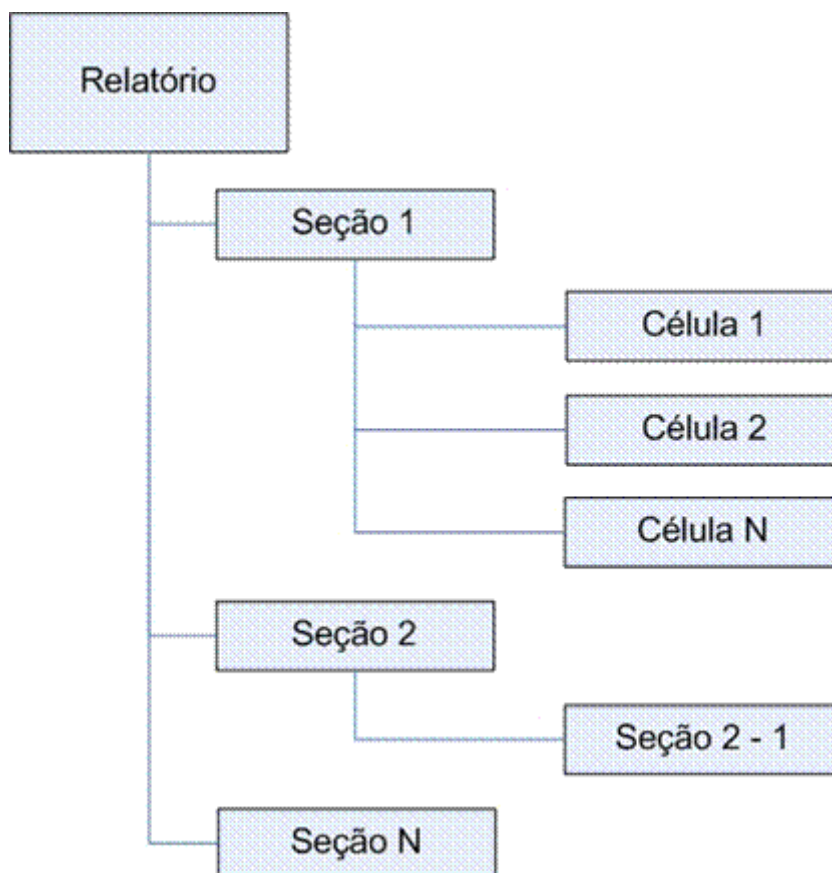
Parametros
 Abrir
 Editar
 Imprimir
 Cancelar

2.3.3.1. Editando o layout do relatório

O primeiro passo é entender a nova estrutura dos relatórios desenvolvidos com a ferramenta TReport.

O Relatório possui Seções e Células. É chamada de Seção, cada um dos grupos de informações, e de Célula, cada um dos campos que serão impressos.

2.3.3.2. Nova estrutura do relatório TReport:



O relatório mais simples que se consegue emitir em TReport, é uma listagem.

2.3.4. Definindo a Função ReportDef()

A função ReportDef() é responsável pela construção do lay-out do relatório (oReport). É ela quem define as colunas, os campos e as informações que serão impressas.

Os comandos que fará essa construção são:

1. DEFINE REPORT
2. DEFINE SECTION
3. DEFINE CELL

DEFINE REPORT

A função DEFINE REPORT é responsável pela criação do objeto Report, ou seja, o relatório. Internamente, o DEFINE REPORT irá executar o método TReport():New().

Estrutura do componente TReport:

- O relatório (TReport) contém 1 ou mais seções (TRSection);

- Uma seção (TRSection) pode conter 1 ou mais seções;
- A seção (TRSection) contém células pré-definidas e células selecionadas pelo usuário;
- A seção (TRSection) também contém as quebras (TRBreak) para impressão de totalizadores (TRFunction);
- Os totalizadores são incluídos pela seção que automaticamente inclui no relatório (TReport).

DEFINE SECTION

Ainda no ReportDef(), são definidas as seções (oSection) do relatório.

As seções do relatório representam os diferentes grupos de informações exibidos.

Há a seção principal e as específicas.

Internamente, o DEFINE SECTION irá executar o método TRSection():New().

A classe TRSection pode ser entendida como um layout do relatório, por conter células, quebras e totalizadores que darão um formato para sua impressão.

Com a classe TRSection é possível definir uma query, filtro ou índice com filtro (IndRegua) que será utilizada por ela para processamento do relatório, através do método Print e utilizando as células de posicionamento (TRPosition).

DEFINE CELL

Para cada seção, devem ser definidas as células. Célula é cada informação que deverá ser impressa.

Pode ser um campo do cadastro, ou um resultado de uma operação. É uma Célula de impressão de uma seção (TRSection) de um relatório que utiliza a classe TReport

Internamente, o DEFINE CELL irá executar o método TRCell():New ().

3. Manipulação de arquivos I

3.1. Geração e leitura de arquivos em formato texto

Arquivos do tipo texto (também conhecidos como padrão TXT) são arquivos com registros de tamanho variável. A indicação do final de cada registro é representada por dois bytes, “0D 0A” em hexadecimal ou “13 10” em decimal ou, ainda, “CR LF” para padrão ASCII.

Apesar do tamanho dos registros ser variável, a maioria dos sistemas gera este tipo de arquivo com registros de tamanho fixo, de acordo com um layout específico que indica quais são os dados gravados.

Para ilustrar estes procedimentos, serão gerados arquivos textos, com duas famílias de funções:

1ª) Família: nesta família serão utilizadas as funções: FCreate(), FWrite(), FClose(), FSeek(), FOpen() e FRead().

2ª) Família: nesta família serão utilizadas as funções: FT_FUse(), FT_FGoTop(), FT_FLastRec(), FT_FEOF(), FT_FReadLn(), FT_FSkip(), FT_FGoto(), FT_FRecno().

Importante

A diferença entre as duas famílias, está na leitura do arquivo texto. Quando se tratar de arquivo texto com tamanho fixo das linhas, poderão ser utilizadas as duas famílias para leitura do arquivo, porém, quando se tratar de arquivo texto com tamanho variável das linhas, somente poderá ser utilizada a segunda família, representada pelas funções: FT_FUse(), FT_FGoto(), FT_FRecno(), FT_FGoTop(), FT_FLastRec(), FT_FEOF(), FT_FReadLn() e FT_FSkip().

3.1.1. 1ª Família de funções de gravação e leitura de arquivos texto

3.1.1.1. FCREATE()

Função de baixo-nível que permite a manipulação direta dos arquivos textos como binários. Ao ser executada FCREATE() cria um arquivo ou elimina o seu conteúdo, e retorna o handle (manipulador) do arquivo, para ser usado nas demais funções de manutenção de arquivo. Após ser utilizado, o Arquivo deve ser fechado através da função FCLOSE().

Na tabela abaixo, estão descritos os atributos para criação do arquivo, definidos no arquivo header fileio.ch

- Atributos definidos no include FileIO.ch

Constante	Valor	Descrição
FC_NORMAL	0	Criação normal do Arquivo (default/padrão).
FC_READONLY	1	Cria o arquivo protegido para gravação.

FC_HIDDEN	2	Cria o arquivo como oculto.
FC_SYSTEM	4	Cria o arquivo como sistema.

Caso desejemos especificar mais de um atributo , basta somá-los . Por exemplo , para criar um arquivo protegido contra gravação e escondido , passamos como atributo FC_READONLY + FC_HIDDEN. .

Nota: Caso o arquivo já exista , o conteúdo do mesmo será ELIMINADO , e seu tamanho será truncado para 0 (ZERO) bytes.

- Sintaxe: FCREATE (< cArquivo > , [nAtributo])
- Parâmetros:

cArquivo	Nome do arquivo a ser criado , podendo ser especificado um path absoluto ou relativo , para criar arquivos no ambiente local (Remote) ou no Servidor, respectivamente .
nAtributo	Atributos do arquivo a ser criado (Vide Tabela de atributos abaixo). Caso não especificado, o DEFAULT é FC_NORMAL.

- Retorno:

Numérico	A função retornará o Handle do arquivo para ser usado nas demais funções de manutenção de arquivo. O Handle será maior ou igual a zero. Caso não seja possível criar o arquivo , a função retornará o handle -1 , e será possível obter maiores detalhes da ocorrência através da função FERROR() .
-----------------	---

3.1.1.2. FWRITE()

Função que permite a escrita em todo ou em parte do conteúdo do buffer , limitando a quantidade de Bytes através do parâmetro nQtyBytes. A escrita começa a partir da posição corrente do ponteiro de arquivos, e a função FWRITE retornará a quantidade real de bytes escritos. Através das funções FOPEN(), FCREATE(), ou FOPENPORT(), podemos abrir ou criar um arquivo ou abrir uma porta de comunicação , para o qual serão gravados ou enviados os dados do buffer informado. Por tratar-se de uma função de manipulação de conteúdo binário , são suportados na String cBuffer todos os caracteres da tabela ASCII , inclusive caracteres de controle (ASC 0 , ASC 12 , ASC 128 , etc.).

Caso aconteça alguma falha na gravação , a função retornará um número menor que o nQtyBytes. Neste caso , a função FERROR() pode ser utilizada para determinar o erro específico ocorrido. A gravação no arquivo é realizada a partir da posição atual do ponteiro , que pode ser ajustado através das funções FSEEK() , FREAD() ou FREADSTR().

- Sintaxe: FWRITE (< nHandle > , < cBuffer > , [nQtyBytes])
- Parâmetros:

nHandle	É o manipulador de arquivo ou device retornado pelas funções FOPEN(), FCREATE(), ou FOPENPORT().
cBuffer	<cBuffer> é a cadeia de caracteres a ser escrita no arquivo especificado. O tamanho desta variável deve ser maior ou igual ao tamanho informado em nQtyBytes (caso seja informado o tamanho).
nQtyBytes	<nQtyBytes> indica a quantidade de bytes a serem escritos a partir da posição corrente do ponteiro de arquivos. Caso seja omitido, todo o conteúdo de <cBuffer> é escrito.

- Retorno:

Numérico	FWRITE() retorna a quantidade de bytes escritos na forma de um valor numérico inteiro. Caso o valor retornado seja igual a <nQtyBytes>, a operação foi bem sucedida. Caso o valor de retorno seja menor que <nBytes> ou zero, ou o disco está cheio ou ocorreu outro erro. Neste caso , utilize a função FERROR() para obter maiores detalhes da ocorrência.
-----------------	--

3.1.1.3. FCLOSE()

Função de tratamento de arquivos de baixo nível utilizada para fechar arquivos binários e forçar que os respectivos buffers do DOS sejam escritos no disco. Caso a operação falhe, FCLOSE() retorna falso (.F.). FERROR() pode então ser usado para determinar a razão exata da falha. Por exemplo, ao tentar-se usar FCLOSE() com um handle (tratamento dado ao arquivo pelo sistema operacional) inválido retorna falso (.F.) e FERROR() retorna erro 6 do DOS, invalid handle. Consulte FERROR() para obter uma lista completa dos códigos de erro.

Nota: Esta função permite acesso de baixo nível aos arquivos e dispositivos do DOS. Ela deve ser utilizada com extremo cuidado e exige que se conheça a fundo o sistema operacional utilizado.

- Sintaxe: FCLOSE (< nHandle >)
- Parâmetros:

nHandle	Handle do arquivo obtido previamente através de FOPEN() ou FCREATE().
----------------	---

- Retorno:

Lógico	Retorna falso (.F.) se ocorre um erro enquanto os buffers estão sendo escritos; do contrário, retorna verdadeiro (.T.).
---------------	---

3.1.1.4. FSEEK()

Função que posiciona o ponteiro do arquivo para as próximas operações de leitura ou gravação. As movimentações de ponteiros são relativas à nOrigem que pode ter os seguintes valores, definidos em fileio.ch:

- Tabela A: Origem a ser considerada para a movimentação do ponteiro de posicionamento do Arquivo.

Origem	Constate(fileio.ch)	Operação
0	FS_SET	Ajusta a partir do início do arquivo. (Default)
1	FS_RELATIVE	Ajuste relativo a posição atual do arquivo.
2	FS_END	Ajuste a partir do final do arquivo.

- Sintaxe: FSEEK (< nHandle > , [nOffset] , [nOrigem])
- Parâmetros:

nHandle	Manipulador obtido através das funções FCREATE,FOPEN.
nOffset	nOffset corresponde ao número de bytes no ponteiro de posicionamento do arquivo a ser movido. Pode ser um numero positivo , zero ou negativo, a ser considerado a partir do parâmetro passado em nOrigem.
nOrigem	Indica a partir de qual posição do arquivo, o nOffset será considerado.

- Retorno:

Numérico	FSEEK() retorna a nova posição do ponteiro de arquivo com relação ao início do arquivo (posição 0) na forma de um valor numérico inteiro. Este valor não leva em conta a posição original do ponteiro de arquivos antes da execução da função FSEEK().
-----------------	--

3.1.1.5. FOPEN()

Função de tratamento de arquivo de baixo nível que abre um arquivo binário existente para que este possa ser lido e escrito, dependendo do argumento <nModo>. Toda vez que houver um erro na abertura do arquivo, FERROR() pode ser usado para retornar o código de erro do Sistema Operacional. Por exemplo, caso o arquivo não exista, FOPEN() retorna -1 e FERROR() retorna 2 para indicar que o arquivo não foi encontrado. Veja FERROR() para uma lista completa dos códigos de erro.

Caso o arquivo especificado seja aberto, o valor retornado é o handle (manipulador) do Sistema Operacional para o arquivo. Este valor é semelhante a um alias no sistema de banco de dados, e ele é exigido para identificar o arquivo aberto para as outras funções de tratamento de arquivo. Portanto, é importante sempre atribuir o valor que foi retornado a uma variável para uso posterior, como mostra o exemplo desta função.

- Sintaxe: FOPEN (< cArq > , [nModo])
- Parâmetros:

cArq	Nome do arquivo a ser aberto que inclui o path caso haja um.
nModo	Modo de acesso DOS solicitado que indica como o arquivo aberto deve ser acessado. O acesso é de uma das categorias relacionadas na tabela A e as restrições de compartilhamento relacionada na Tabela B. O modo padrão é zero, somente para leitura, com compartilhamento por Compatibilidade. Ao definirmos o modo de acesso , devemos somar um elemento da Tabela A com um elemento da Tabela B.

- Retorno:

Numérico	FOPEN() retorna o handle de arquivo aberto na faixa de zero a 65.535. Caso ocorra um erro, FOPEN() retorna -1.
----------	--

3.1.1.6. FREAD()

Função que realiza a leitura dos dados a partir um arquivo aberto, através de FOPEN(), FCREATE() e/ou FOPENPORT(), e armazena os dados lidos por referência no buffer informado.

FREAD() lerá até o número de bytes informado em nQtyBytes; caso aconteça algum erro ou o arquivo chegue ao final, FREAD() retornará um número menor que o especificado em nQtyBytes. FREAD() lê normalmente caracteres de controle (ASC 128, ASC 0, etc.) e lê a partir da posição atual do ponteiro atual do arquivo , que pode ser ajustado ou modificado pelas funções FSEEK() , FWRITE() ou FREADSTR().

A variável String a ser utilizada como buffer de leitura deve ser sempre pré-alocado e passado como referência. Caso contrário, os dados não poderão ser retornados.

- Sintaxe: FREAD (< nHandle > , < cBuffer > , < nQtyBytes >)
- Parâmetros:

nHandle	É o manipulador (Handle) retornado pelas funções FOPEN(), FCREATE(), FOPENPORT(), que faz referência ao arquivo a ser lido.
---------	---

cBuffer	É o nome de uma variável do tipo String , a ser utilizada como buffer de leitura , onde os dados lidos deverão ser armazenados. O tamanho desta variável deve ser maior ou igual ao tamanho informado em nQtyBytes. Esta variável deve ser sempre passada por referência. (@ antes do nome da variável), caso contrário os dados lidos não serão retornados.
nQtyBytes	Define a quantidade de Bytes que devem ser lidos do arquivo a partir posicionamento do ponteiro atual.

- Retorno:

Numérico	Quantidades de bytes lidos. Caso a quantidade seja menor que a solicitada, isto indica erro de leitura ou final de arquivo, Verifique a função FERROR() para maiores detalhes.
-----------------	--

Exemplo: Geração de arquivo TXT, utilizando a primeira família de funções

```
#include "protheus.ch"
```

```
/*
```

```
+-----
```

```
| Programa | GeraTXT | Autor | SERGIO FUZINAKA | Data
```

```
+-----
```

```
| Descrição | Gera o arquivo TXT, a partir do Cadastro de Clientes |
```

```
+-----
```

```
| Uso | Curso ADVPL
```

```
+-----
```

```
*/
```

```
User Function GeraTXT()
```

```
//+-----+
```

```
//| Declaração de Variáveis |
```

```
//+-----+
```

```
Local oGeraTxt
```

```
Private cPerg := "EXPSA1"
```

```
Private cAlias := "SA1"
```

```
//CriaSx1(cPerg)
```

```
//Pergunte(cPerg,.F.)
dbSelectArea(cAlias)
dbSetOrder(1)
```

```
//+-----+
//| Montagem da tela de processamento. |
//+-----+
```

```
DEFINE MSDIALOG oGeraTxt TITLE OemToAnsi("Geração de Arquivo Texto") ;
FROM 000,000 TO 200,400 PIXEL
```

```
@ 005,005 TO 095,195 OF oGeraTxt PIXEL
@ 010,020 Say " Este programa ira gerar um arquivo texto, conforme os parame- ";
OF oGeraTxt PIXEL
@ 018,020 Say " tros definidos pelo usuário, com os registros do arquivo de ";
OF oGeraTxt PIXEL
@ 026,020 Say " SA1 " OF oGeraTxt PIXEL
```

```
DEFINE SBUTTON FROM 070, 030 TYPE 1 ;
ACTION (OkGeraTxt(),oGeraTxt:End()) ENABLE OF oGeraTxt
```

```
DEFINE SBUTTON FROM 070, 070 TYPE 2 ;
ACTION (oGeraTxt:End()) ENABLE OF oGeraTxt
```

```
DEFINE SBUTTON FROM 070, 110 TYPE 5 ;
ACTION (Pergunte(cPerg,.T.)) ENABLE OF oGeraTxt
```

```
ACTIVATE DIALOG oGeraTxt CENTERED
```

```
Return Nil
```

```
/*
+-----+
| Função   | OKGERATXT | Autor | SERGIO FUZINAKA | Data |
+-----+
| Descrição | Função chamada pelo botão OK na tela inicial de processamento. |
|           | Executa a geração do arquivo texto.           |
```

```

+-----
/*/

Static Function OkGeraTxt

//+-----
//| Cria o arquivo texto
//+-----
Private cArqTxt := "\SYSTEM\EXPSA1.TXT"
Private nHdl := fCreate(cArqTxt)

If nHdl == -1
    MsgAlert("O arquivo de nome "+cArqTxt+" não pode ser executado! Verifique os
    parâmetros.", "Atenção!")
    Return
Endif

// Inicializa a régua de processamento
Processa({| | RunCont() }, "Processando...")

Return Nil

/*/
+-----
| Função | RUNCONT | Autor | SERGIO FUZINAKA | Data |
+-----
| Descrição | Função auxiliar chamada pela PROCESSA. A função PROCESSA | |
| | | monta a janela com a régua de processamento. |
| | |
+-----
/*/

Static Function RunCont

Local cLin

dbSelectArea(cAlias)

```

```

dbGoTop()
ProcRegua(RecCount()) // Numero de registros a processar

While (cAlias)->(!EOF())
//Incrementa a régua
IncProc()

cLin := (cAlias)->A1_FILIAL
cLin += (cAlias)->A1_COD
cLin += (cAlias)->A1_LOJA
        cLin += (cAlias)->A1_NREDUZ
        cLin += STRZERO((cAlias)->A1_MCOMPRA*100,16) // 14,2
        cLin += DTOS((cAlias)->A1_ULTCOM)//AAAAMMDD
        cLin += CRLF

//+-----+
//| Gravação no arquivo texto. Testa por erros durante a gravação da |
//| linha montada.                                     |
//+-----+

If fWrite(nHdl,cLin,Len(cLin)) != Len(cLin)
                                If !MsgAlert("Ocorreu um erro na gravação do
arquivo."+
                                "Continua?","Atenção!")
                                Exit
                                Endif
        Endif

        (cAlias)->(dbSkip())
EndDo

// O arquivo texto deve ser fechado, bem como o dialogo criado na função anterior
fClose(nHdl)

Return Nil

```

Note que para a geração do arquivo TXT foram utilizadas, basicamente, as funções FCreate, FWrite e FClose que, respectivamente, gera o arquivo, adiciona dados e fecha o arquivo. No exemplo, o formato é estabelecido pela concatenação dos dados na variável cLin a qual é utilizada na gravação dos dados. Para a leitura de dados TXT serão utilizada as funções FOpen e FRead.

Exemplo: Leitura de arquivo TXT, utilizando a primeira família de funções

```
#Include "protheus.ch"

/*
+-----+
| Programa | LeTXT                                | Autor | SERGIO FUZINAKA | Data |
|          |                                     |      |                   |     |
+-----+
| Descrição | Leitura de arquivo TXT                                |
+-----+
| Uso       | Curso ADVPL                                           |
+-----+
*/
User Function LeTXT()

//+-----+
//| Declaração de Variáveis                                |
//+-----+

Local cPerg := "IMPSA1"
Local oLeTxt

Private cAlias := "SA1"

//CriaSx1(cPerg)
//Pergunte(cPerg,.F.)

dbSelectArea(cAlias)
dbSetOrder(1)
//+-----+
// Montagem da tela de processamento
```



```
//+-----+
```

```
DEFINE MSDIALOG oLeTxt TITLE OemToAnsi("Leitura de Arquivo Texto");
FROM 000,000 TO 200,400 PIXEL
@ 005,005 TO 095,195 OF oLeTxt PIXEL
@ 10,020 Say " Este programa ira ler o conteúdo de um arquivo texto, conforme";
OF oLeTxt PIXEL
@ 18,020 Say " os parâmetros definidos pelo usuário, com os registros do arquivo";
OF oLeTxt PIXEL
@ 26,020 Say " SA1" OF oLeTxt PIXEL
```

```
DEFINE SBUTTON FROM 070, 030 TYPE 1 ;
ACTION (OkLeTxt(),oLeTxt:End()) ENABLE OF oLeTxt
```

```
DEFINE SBUTTON FROM 070, 070 TYPE 2 ;
ACTION (oLeTxt:End()) ENABLE OF oLeTxt
```

```
DEFINE SBUTTON FROM 070, 110 TYPE 5 ;
ACTION (Pergunte(cPerg,.T.)) ENABLE OF oLeTxt
ACTIVATE DIALOG oLeTxt CENTERED
```

```
Return Nil
```

```
/*/
```

```
+-----
```

Função	OKLETXT	Autor	SERGIO FUZINAKA	Data
--------	---------	-------	-----------------	------

```
+-----
```

Descrição	Função chamada pelo botão OK na tela inicial de processamento
	Executa a leitura do arquivo texto

```
+-----
```

```
/*/
```

```
Static Function OkLeTxt()
```

```
//+-----+
```

```

//| Abertura do arquivo texto
//+-----+

Private cArqTxt := "\SYSTEM\EXPSA1.TXT"
Private nHdl := fOpen(cArqTxt,68)

If nHdl == -1
    MsgAlert("O arquivo de nome "+cArqTxt+" não pode ser aberto! Verifique os
    parâmetros.", "Atenção!")
    Return
Endif

// Inicializa a régua de processamento
Processa({ | RunCont() }, "Processando...")
Return Nil

/*
+-----+
| Função | RUNCONT | Autor | SERGIO FUZINAKA | Data |
|
+-----+
| Descrição | Função auxiliar chamada pela PROCESSA. A função PROCESSA |
| | monta a janela com a régua de processamento. |
|
+-----+
*/

Static Function RunCont

Local nTamFile := 0
Local nTamLin := 56
Local cBuffer := ""
Local nBtLidos := 0
Local cFilSA1 := ""
Local cCodSA1 := ""
Local cLojaSA1 := ""

```

```
//123456789012345678901234567890123456789012345678901234567890
//000000000010000000002000000000300000000040000000005000000000600000000070
//FFCCCCCLNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNVVVVVVVVVVVVVVVVVVDDDDDDDD
//A1_FILIAL - 01, 02 - TAM: 02
//A1_COD - 03, 08 - TAM: 06
//A1_LOJA - 09, 10 - TAM: 02
//A1_NREDUZ - 11, 30 - TAM: 20
//A1_MCOMPRA - 31, 46 - TAM: 14,2
//A1_ULTCOM - 47, 54 - TAM: 08
```

```
nTamFile := fSeek(nHdl,0,2)
```

```
fSeek(nHdl,0,0)
```

```
cBuffer := Space(nTamLin) // Variável para criação da linha do registro para leitura
```

```
ProcRegua(nTamFile) // Numero de registros a processar
```

```
While nBtLidos < nTamFile
```

```
//Incrementa a régua
```

```
IncProc()
```

```
// Leitura da primeira linha do arquivo texto
```

```
nBtLidos += fRead(nHdl,@cBuffer,nTamLin)
```

```
cFilSA1 := Substr(cBuffer,01,02) //- 01, 02 - TAM: 02
```

```
cCodSA1 := Substr(cBuffer,03,06) //- 03, 08 - TAM: 06
```

```
cLojaSA1 := Substr(cBuffer,09,02) //- 09, 10 - TAM: 02
```

```
While .T.
```

```
IF dbSeek(cFilSA1+cCodSA1+cLojaSA1)
```

```
cCodSA1 := SOMA1(cCodSA1)
```

```
Loop
```

```
Else
```

```
Exit
```

```
Endif
```

```
Enddo
```

```
dbSelectArea(cAlias)
```

```

RecLock(cAlias,.T.)
(cAlias)->A1_FILIAL      := cFilSA1      //- 01, 02 - TAM: 02
(cAlias)->A1_COD          := cCodSA1      //- 03, 08 - TAM: 06
(cAlias)->A1_LOJA         := cLojaSA1     //- 09, 10 - TAM: 02
(cAlias)->A1_NREDUZ       := Substr(cBuffer,11,20)
//- 11, 30 - TAM: 20
(cAlias)->A1_MCOMPRA      := Val(Substr(cBuffer,31,16))/100
//- 31, 46 - TAM: 14,2
(cAlias)->A1_ULTCOM       := STOD(Substr(cBuffer,47,08))
//- 47, 54 - TAM: 08
MSUnlock()

EndDo

// O arquivo texto deve ser fechado, bem como o dialogo criado na função anterior.
fClose(nHdl)

Return Nil

```

3.1.2. 2ª Família de funções de gravação e leitura de arquivos texto

3.1.2.1. FT_FUSE()

Função que abre ou fecha um arquivo texto para uso das funções FT_F*. As funções FT_F* são usadas para ler arquivos texto, onde as linhas são delimitadas pela sequência de caracteres CRLF ou LF (*) e o tamanho máximo de cada linha é 1022 bytes.. O arquivo é aberto em uma área de trabalho, similar à usada pelas tabelas de dados.

- Sintaxe: FT_FUSE ([cTXTFile])
- Parâmetros:

cTXTFile	Corresponde ao nome do arquivo TXT a ser aberto. Caso o nome não seja passado, e já exista um arquivo aberto. o mesmo é fechado.
-----------------	--

- Retorno:

Numérico	A função retorna o Handle de controle do arquivo. Em caso de falha de abertura, a função retornará -1
-----------------	---

3.1.2.2. FT_FGOTOP()

A função tem como objetivo mover o ponteiro, que indica a leitura do arquivo texto, para a posição absoluta especificada pelo argumento <nPos>.

- Sintaxe: FT_FGOTO (< nPos >)
- Parâmetros:

nPos	Indica a posição que será colocado o ponteiro para leitura dos dados no arquivo.
-------------	--

3.1.2.3. FT_FLASTREC()

Função que retorna o número total de linhas do arquivo texto aberto pela FT_FUse. As linhas são delimitadas pela seqüência de caracteres CRLF o LF.

- Sintaxe: FT_FLASTREC()
- Parâmetros:

Nenhum	.
---------------	---

- Retorno:

Numérico	Retorna a quantidade de linhas existentes no arquivo. Caso o arquivo esteja vazio, ou não exista arquivo aberto, a função retornará 0 (zero).
-----------------	---

3.1.2.4. FT_FEOF()

Função que retorna verdadeiro (.t.) se o arquivo texto aberto pela função FT_FUSE() estiver posicionado no final do arquivo, similar à função EOF() utilizada para arquivos de dados.

- Sintaxe: FT_FEOF()
- Parâmetros:

Nenhum	.
---------------	---

- Retorno:

Lógico	Retorna true caso o ponteiro do arquivo tenha chegado ao final, false caso contrário.
---------------	---

3.1.2.5. FT_FREADLN()

Função que retorna uma linha de texto do arquivo aberto pela FT_FUSE. As linhas são delimitadas pela sequência de caracteres CRLF (chr(13) + chr(10)), ou apenas LF (chr(10)), e o tamanho máximo de cada linha é 1022 bytes.

- Sintaxe: FT_FREADLN()
- Parâmetros:

Nenhum	.
---------------	---

- Retorno:

Caracter	Retorna a linha inteira na qual está posicionado o ponteiro para leitura de dados.
-----------------	--

3.1.2.6. FT_FSKIP()

Função que move o ponteiro do arquivo texto aberto pela FT_FUSE() para a próxima linha, similar ao DBSKIP() usado para arquivos de dados.

- Sintaxe: FT_FSKIP ([nLinhas])
- Parâmetros:

nLinhas	nLinhas corresponde ao número de linhas do arquivo TXT ref. movimentação do ponteiro de leitura do arquivo.
----------------	---

- Retorno

Nenhum	.
---------------	---

3.1.2.7. FT_FGOTO()

Função utilizada para mover o ponteiro, que indica a leitura do arquivo texto, para a posição absoluta especificada pelo argumento <nPos>.

- Sintaxe: FT_FGOTO (< nPos >)
- Parâmetros:

nPos	Indica a posição que será colocado o ponteiro para leitura dos dados no arquivo.
-------------	--

- Retorno:

Nenhum	.
---------------	---

3.1.2.8. FT_FRECNO()

A função tem o objetivo de retornar a posição do ponteiro do arquivo texto.

A função FT_FRecno retorna a posição corrente do ponteiro do arquivo texto aberto pela FT_FUse.

- Sintaxe: FT_FRECNO ()
- Parâmetros:

Nenhum	.
--------	---

- Retorno:

Caracter	Retorna a posição corrente do ponteiro do arquivo texto.
----------	--

Exemplo: Leitura de arquivo TXT, utilizando a segunda família de funções

```
#Include "Protheus.ch"
```

```
/*/
```

```
+-----+
| Programa | LeArqTXT                | Autor | Robson Luiz          | Data |
|                                     |      |                          |      |
+-----+
```

```
+-----+
| Descrição | Leitura de arquivo TXT          |
+-----+
```

```
+-----+
| Uso      | Curso ADVPL
|          |
+-----+
```

```
+-----/*/
```

```
User Function LeArqTxt()
```

```
Private nOpc                := 0
Private cCadastro           := "Ler arquivo texto"
Private aSay                := {}
Private aButton             := {}
```

```
AADD( aSay, "O objetivo desta rotina e efetuar a leitura em um arquivo texto" )
```

```
AADD( aButton, { 1,.T.,{| | nOpc := 1,FecharBatch()}})
```

```
AADD( aButton, { 2,.T.,{| | FecharBatch() } } )
```

```
FormBatch( cCadastro, aSay, aButton )
```

```
If nOpc == 1
```

```
    Processa( { | | Import() }, "Processando..." )
```

```
Endif
```

```
Return Nil
```

```
//+-----
```

```
//| Função - Import()
```

```
//+-----
```

```
Static Function Import()
```

```
Local cBuffer                := ""
```

```
Local cFileOpen              := ""
```

```
Local cTitulo1               := "Selecione o arquivo"
```

```
Local cExtens                 := "Arquivo TXT | *.txt"
```

```
/**
```

```
*
```

```
* cGetFile(<ExpC1>,<ExpC2>,<ExpN1>,<ExpC3>,<ExpL1>,<ExpN2>)
```

```
* -----
```

```
* <ExpC1> - Expressão de filtro
```

```
* <ExpC2> - Titulo da janela
```

```
* <ExpN1> - Numero de mascara default 1 para *.Exe
```

```
* <ExpC3> - Diretório inicial se necessário
```

```
* <ExpL1> - .F. botão salvar - .T. botão abrir
```

```
* <ExpN2> - Mascara de bits para escolher as opções de visualização do objeto
```

```
* (prconst.ch)
```

```
*/
```

```
cFileOpen := cGetFile(cExtens,cTitulo1,,cMainPath,.T.)
```

```
If !File(cFileOpen)
```

```
    MsgAlert("Arquivo texto: "+cFileOpen+" não localizado",cCadastro)
```

```
    Return
```

```
Endif
```

```
FT_FUSE(cFileOpen) //ABRIR
```



```
FT_FGOTOP() //PONTO NO TOPO
ProcRegua(FT_FLASTREC()) //QTOS REGISTROS LER

While !FT_FEOF() //FACA ENQUANTO NAO FOR FIM DE ARQUIVO
  IncProc()

  // Capturar dados
  cBuffer := FT_FREADLN() //LENDO LINHA

  cMsg := "Filial: " + SubStr(cBuffer,01,02) + Chr(13)+Chr(10)
  cMsg += "Código: " + SubStr(cBuffer,03,06) + Chr(13)+Chr(10)
  cMsg += "Loja: " + SubStr(cBuffer,09,02) + Chr(13)+Chr(10)
  cMsg += "Nome fantasia: " + SubStr(cBuffer,11,15) + Chr(13)+Chr(10)
  cMsg += "Valor: " + SubStr(cBuffer,26,14) + Chr(13)+Chr(10)
  cMsg += "Data: " + SubStr(cBuffer,40,08) + Chr(13)+Chr(10)

  MsgInfo(cMsg)

  FT_FSKIP() //próximo registro no arquivo txt
EndDo
Exemplo (continuação):

FT_FUSE() //fecha o arquivo txt
MsgInfo("Processo finalizada")
Return Nil
```

Exercício

Desenvolver uma rotina que realize a exportação dos itens marcados no cadastro de clientes para um arquivo TXT em um diretório especificado pelo usuário.

Exercício

Desenvolver uma rotina que realize a importação dos dados de clientes contidos em um arquivo TXT, sendo o mesmo selecionado pelo usuário.

4. Oficina de programação I

4.1. Interfaces com sintaxe clássica

A sintaxe convencional para definição de componentes visuais da linguagem ADVPL depende diretamente no include especificado no cabeçalho do fonte. Os dois includes disponíveis para o ambiente ADVPL Protheus são:

- RWMAKE.CH: permite a utilização da sintaxe CLIPPER na definição dos componentes visuais.
- PROTHEUS.CH: permite a utilização da sintaxe ADVPL convencional, a qual é um aprimoramento da sintaxe CLIPPER, com a inclusão de novos atributos para os componentes visuais disponibilizados no ERP Protheus.

Para ilustrar a diferença na utilização destes dois includes, segue abaixo as diferentes definições para o componentes Dialog e MsDialog:

Exemplo 01 – Include Rwmake.ch

```
#include "rwmake.ch"

@ 0,0 TO 400,600 DIALOG oDlg TITLE "Janela em sintaxe Clipper"
ACTIVATE DIALOG oDlg CENTERED
```

Exemplo 02 – Include Protheus.ch

```
#include "protheus.ch"

DEFINE MSDIALOG oDlg TITLE "Janela em sintaxe ADVPL "FROM 000,000 TO 400,600 PIXEL
ACTIVATE MSDIALOG oDlg CENTERED
```

Importante

Ambas as sintaxes produzirão o mesmo efeito quando compiladas e executadas no ambiente Protheus, mas deve ser utilizada sempre a sintaxe ADVPL através do uso do include PROTHEUS.CH

Os componentes da interface visual que serão tratados neste tópico, utilizando a sintaxe clássica da linguagem ADVPL são:

- ☐ BUTTON()
- ☐ CHECKBOX()
- ☐ COMBOBOX()
- ☐ FOLDER()

- MSDIALOG()
- MSGET()
- RADIO()
- SAY()
- SBUTTON()

Executar o fonte **DIALOG_OBJETOS.PRW** e avaliar a definição dos componentes utilizados utilizando a sintaxe clássica.

BUTTON()

Sintaxe	@ nLinha,nColuna BUTTON cTexto SIZE nLargura,nAltura UNIDADE OF oObjetoRef ACTION AÇÃO
Descrição	Define o componente visual Button, o qual permite a inclusão de botões de operação na tela da interface, os quais serão visualizados somente com um texto simples para sua identificação.

CHECKBOX()

Sintaxe	@ nLinha,nColuna CHECKBOX oCheckBox VAR VARIABEL PROMPT cTexto WHEN WHEN UNIDADE OF oObjetoRef SIZE nLargura,nAltura MESSAGE cMensagem
Descrição	Define o componente visual CheckBox, o qual permite a utilização da uma marca para habilitar ou não uma opção escolhida, sendo esta marca acompanhada de um texto explicativo. Difere do RadioMenu pois cada elemento do check é único, mas o Radio permite a utilização de uma lista junto com um controle de seleção.

COMBOBOX()

Sintaxe	@ nLinha,nColuna COMBOBOX VARIABEL ITEMS AITENS SIZE nLargura,nAltura UNIDADE OF oObjetoRef
Descrição	Define o componente visual ComboBox, o qual permite seleção de um item dentro de uma lista de opções de textos simples no formato de um vetor.

FOLDER()

Sintaxe	@ nLinha,nColuna FOLDER oFolder OF oObjetoRef PROMPT &cTexto1,...,&cTextoX PIXEL SIZE nLargura,nAltura
Descrição	Define o componente visual Folder, o qual permite a inclusão de diversos Dialogs dentro de uma mesma interface visual. Um Folder pode ser entendido como um array de Dialogs, aonde cada painel recebe seus componentes e tem seus atributos definidos independentemente dos demais.

MSDIALOG()

Sintaxe	DEFINE MSDIALOG oObjetoDLG TITLE cTitulo FROM nLinIni,nColIni TO nLiFim,nColFim OF oObjetoRef UNIDADE
Descrição	Define o componente MSDIALOG(), o qual é utilizado como base para os demais componentes da interface visual, pois um componente MSDIALOG() é uma janela da aplicação.

MSGET()

Sintaxe	@ nLinha, nColuna MSGET VARIABEL SIZE nLargura,nAltura UNIDADE OF oObjetoRef F3 cF3 VALID VALID WHEN WHEN PICTURE cPicture
Descrição	Define o componente visual MSGET, o qual é utilizado para captura de informações digitáveis na tela da interface.

RADIO()

Sintaxe	@ nLinha,nColuna RADIO oRadio VAR nRadio 3D SIZE nLargura,nAltura <ITEMS PROMPT> cItem1,cItem2,...,cItemX OF oObjetoRef UNIDADE ON CHANGE CHANGE ON CLICK CLICK
Descrição	Define o componente visual Radio, também conhecido como RadioMenu, o qual é seleção de uma opção ou de múltiplas opções através de uma marca para os itens exibidos de uma lista. Difere do componente CheckBox, pois cada elemento de check é sempre único, e o Radio pode conter um ou mais elementos.

SAY()

Sintaxe	@ nLinha, nColuna SAY cTexto SIZE nLargura,nAltura UNIDADE OF oObjetoRef
Descrição	Define o componente visual SAY, o qual é utilizado para exibição de textos em uma tela de interface.

SBUTTON()

Sintaxe	DEFINE SBUTTON FROM nLinha, nColuna TYPE N ACTION AÇÃO STATUS OF oObjetoRef
Descrição	Define o componente visual SButton, o qual permite a inclusão de botões de operação na tela da interface, os quais serão visualizados dependendo da interface do sistema ERP utilizada somente com um texto simples para sua identificação, ou com uma imagem (BitMap) pré-definido.

4.2. Régua de processamento

Os indicadores de progresso ou régua de processamento disponíveis na linguagem ADVPL que serão abordados neste material são:

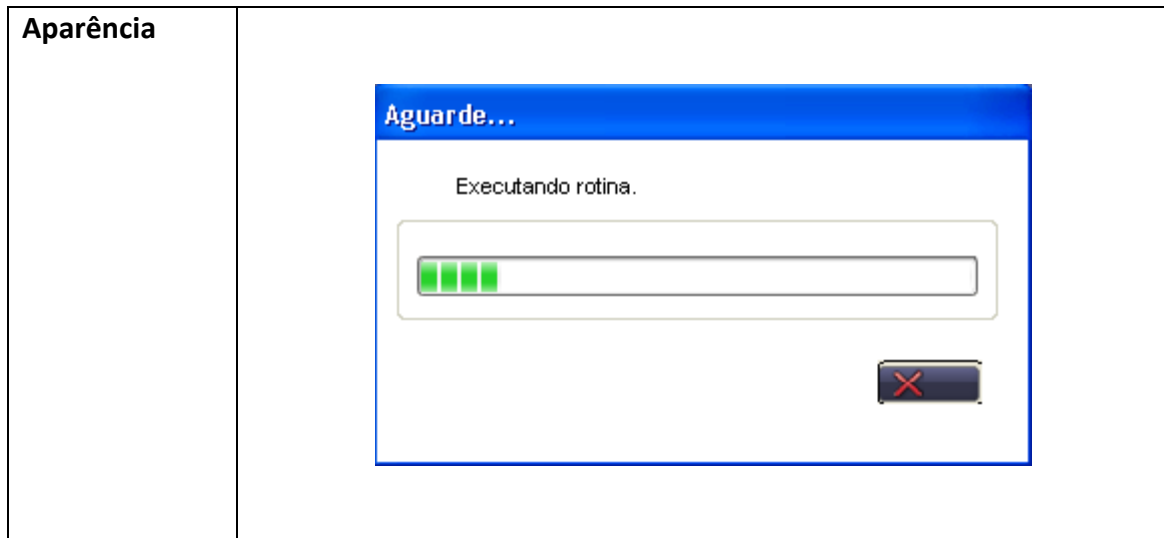
- ❏ RPTSTATUS()
- ❏ PROCESSA()
- ❏ MSNEWPROCESS()
- ❏ MSAGUARDE()
- ❏ MSGRUN()

4.2.1. RptStatus()

Régua de processamento simples, com apenas um indicador de progresso, utilizada no processamento de relatórios do padrão SetPrint().

- Sintaxe: RptStatus(bAcao, cMensagem)
- Retorno: Nil
- Parâmetros:

bAcao	Bloco de código que especifica a ação que será executada com o acompanhamento da régua de processamento.
cMensagem	Mensagem que será exibida na régua de processamento durante a execução.



Exemplo: Função RPTStatus() e acessórias

```
/*/
```

```
+-----
```

```
| Função   | GRPTSTATUS | Autor | ROBSON LUIZ           | Data |      |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização das funções RPTSTATUS() |
```

```
|           | SETREGUA() E INCREGUA() |
```

```
|
```

```
+-----
```

```
| Uso       | Curso ADVPL |
```

```
+-----
```

```
/*/
```

```
User Function GRptStatus()
```

```
Local aSay   := {}
```

```
Local aButton := {}
```

```
Local nOpc   := 0
```

```
Local cTitulo := "Exemplo de Funções"
```

```
Local cDesc1  := "Este programa exemplifica a utilização da função Processa() em conjunto"
```

```
Local cDesc2  := "com as funções de incremento ProcRegua() e IncProc()"
```

```
Private cPerg := "RPTSTA"
```

```
CriaSX1()
```

```
Pergunte(cPerg,.F.)
```

```
AADD( aSay, cDesc1 )
```

```
AADD( aSay, cDesc2 )
```

```
AADD( aButton, { 5, .T., { || Pergunte(cPerg,.T. ) } } )
```

```
AADD( aButton, { 1, .T., { || nOpc := 1, FechaBatch() } } )
```

```
AADD( aButton, { 2, .T., { || FechaBatch() } } )
```

```
FormBatch( cTitulo, aSay, aButton )
```

```
If nOpc <> 1
```

```
    Return Nil
```

```
Endif
```

```
RptStatus( { || End | RunProc(@lEnd)}, "Aguarde...", "Executando rotina.", .T. )
```

```
Return Nil
```

Exemplo: Funções acessórias da RPTStatus()

```
/*/
```

```
+-----
```

Função	RUNPROC	Autor	ROBSON LUIZ	Data
--------	---------	-------	-------------	------

```
+-----
```

Descrição	Função de processamento executada através da RPTSTATUS()			
-----------	--	--	--	--

```
+-----
```

Uso	Curso ADVPL
-----	-------------

```
|
```

```
+-----
```

```
/*/
```

```
Static Function RunProc(lEnd)
```

```
Local nCnt := 0
```

```
dbSelectArea("SX5")
```

```
dbSetOrder(1)
```

```
dbSeek(xFilial("SX5")+mv_par01,.T.)
```

```
While !Eof() .And. X5_FILIAL == xFilial("SX5") .And. X5_TABELA <= mv_par02
  nCnt++
  dbSkip()
End
```

```
dbSeek(xFilial("SX5")+mv_par01,.T.)
```

```
SetRegua(nCnt)
While !Eof() .And. X5_FILIAL == xFilial("SX5") .And. X5_TABELA <= mv_par02
  IncRegua()
  If !End
    MsgInfo(cCancel,"Fim")
    Exit
  Endif
  dbSkip()
End
Return .T.
```

SETREGUA()

A função SetRegua() é utilizada para definir o valor máximo da régua de progressão criada através da função RptStatus().

- Sintaxe: SetRegua(nMaxProc)
- Parâmetros:

nMaxProc	Variável que indica o valor máximo de processamento (passos) que serão indicados pela régua.
-----------------	--

- Retorno:

Nenhum	.
---------------	---

Exemplo:

...

```
dbSelectArea("SA1")
```



```
dbGoTop()
SetRegua>LastRec())
While !Eof()
    IncRegua()
    If Li > 60
...

```

INCREGUA()

A função IncRegua() é utilizada para incrementar valor na régua de progressão criada através da função RptStatus()

- Sintaxe: IncRegua(cMensagem)
- Parâmetros:

cMensagem	Mensagem que será exibida e atualizada na régua de processamento a cada execução da função IncRegua(), sendo que a taxa de atualização da interface é controlada pelo Binário.
------------------	--

- Retorno:

Nenhum	.
---------------	---

Exemplo:

```
...

dbSelectArea("SA1")
dbGoTop()
SetRegua>LastRec())
While !Eof()
    IncRegua("Avaliando cliente:" + SA1->A1_COD)
    If Li > 60
...


```

4.2.2. Processa()

Régua de processamento simples, com apenas um indicador de progresso, utilizada no processamento de rotinas.

- Sintaxe: Processa(bAcao, cMensagem)

- Retorno: Nil
- Parâmetros:

bAcao	Bloco de código que especifica a ação que será executada com o acompanhamento da régua de processamento.
cMensagem	Mensagem que será exibida na régua de processamento durante a execução.
Aparência	

Exemplo: Função PROCESSA() e acessórias

```
/*/
```

```
+-----
```

```
| Função   | GPROCES1 | Autor | ROBSON LUIZ   | Data |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização das funções PROCESSA() |
```

```
|                                     | PROCREGUA() E INCPROC() |
```

```
|
```

```
+-----
```

```
| Uso       | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
User Function GProces1()
```

```
Local aSay := {}
```

```

Local aButton := {}
Local nOpc := 0
Local cTitulo := "Exemplo de Funções"
Local cDesc1 := "Este programa exemplifica a utilização da função Processa()"
Local cDesc2 := " em conjunto com as funções de incremento ProcRegua() e"
Local cDesc3 := " IncProc()"

Private cPerg := "PROCES"

CriaSX1()
Pergunte(cPerg,.F.)

AADD( aSay, cDesc1 )
AADD( aSay, cDesc2 )

AADD( aButton, { 5, .T., { | | Pergunte(cPerg,.T. ) } } )
AADD( aButton, { 1, .T., { | | nOpc := 1, FechaBatch() } } )
AADD( aButton, { 2, .T., { | | FechaBatch() } } )

FormBatch( cTitulo, aSay, aButton )

If nOpc <> 1
    Return Nil
Endif

Processa( { | | End | RunProc(@|End)}, "Aguarde...", "Executando rotina.", .T. )

Return Nil

/*/
+-----+
| Função   | RUNPROC   | Autor | ROBSON LUIZ   | Data |           |
+-----+
| Descrição | Função de processamento executada através da  PROCRSSA() |
+-----+
| Uso       | Curso ADVPL
|

```

```

+-----
/*/

Static Function RunProc(IEnd)
Local nCnt := 0

dbSelectArea("SX5")
dbSetOrder(1)
dbSeek(xFilial("SX5")+mv_par01,.T.)

dbEval( { |x| nCnt++ },,{ |X5_FILIAL==xFilial("SX5").And.X5_TABELA<=mv_par02})

dbSeek(xFilial("SX5")+mv_par01,.T.)

ProcRegua(nCnt)
While !Eof() .And. X5_FILIAL == xFilial("SX5") .And. X5_TABELA <= mv_par02
    IncProc("Processando tabela: "+SX5->X5_CHAVE)
    If IEnd
        MsgInfo(cCancela,"Fim")
        Exit
    Endif
    dbSkip()
End
Return .T.

```

SETPROC()

A função SetProc() é utilizada para definir o valor máximo da régua de progressão criada através da função Processa().

- Sintaxe: Processa(nMaxProc)
- Parâmetros:

nMaxProc	Variável que indica o valor máximo de processamento (passos) que serão indicados pela régua.
-----------------	--

- Retorno:

Nenhum	.
---------------	---

Exemplo:

```
...  
dbSelectArea("SA1")  
dbGoTop()  
SetProc(LastRec())  
While !Eof()  
    IncProc()  
    If Li > 60  
...  

```

INCPROC()

A função IncProc() é utilizada para incrementar valor na régua de progressão criada através da função Processa()

- Sintaxe: IncProc(cMensagem)
- Parâmetros:

cMensagem	Mensagem que será exibida e atualizada na régua de processamento a cada execução da função IncProc(), sendo que a taxa de atualização da interface é controlada pelo Binário.
------------------	---

- Retorno:

Nenhum	.
---------------	---


Exemplo:

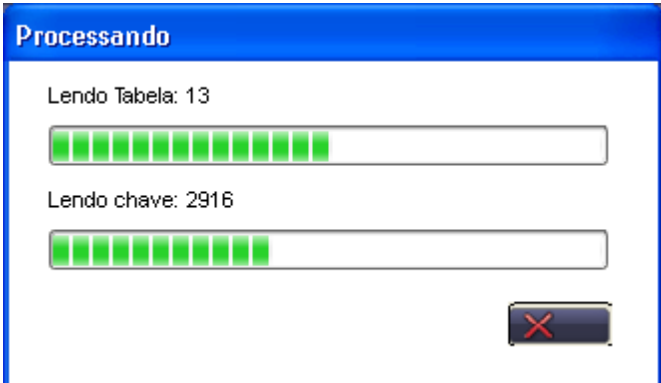
```
...  
dbSelectArea("SA1")  
dbGoTop()  
SetProc(LastRec())  
While !Eof()  
    IncProc("Avaliando cliente:"+SA1->A1_COD)  
    If Li > 60  
...  

```

4.2.3. MsNewProcess().

Régua de processamento dupla, possuindo dois indicadores de progresso independentes, utilizada no processamento de rotinas.

- Sintaxe: MsNewProcess():New(bAcao, cMensagem)
- Retorno: oProcess  objeto do tipo MsNewProcess()
- Parâmetros:

bAcao	Bloco de código que especifica a ação que será executada com o acompanhamento da régua de processamento.
cMensagem	Mensagem que será exibida na régua de processamento durante a execução.
Aparência	

- Métodos:

Activate()	Inicia a execução do objeto MsNewProcess instanciado.
SetRegua1()	Define a quantidade de informações que serão demonstradas pelo indicador de progresso superior. Parâmetro: nMaxProc
IncRegua1()	Incrementa em uma unidade o indicador de progresso superior, o qual irá demonstrar a evolução do processamento de acordo com a quantidade definida pelo método SetRegua1(). Parâmetro: cMensagem
SetRegua2()	Define a quantidade de informações que serão demonstradas pelo indicador de progresso inferior. Parâmetro: nMaxProc
IncRegua2()	Incrementa em uma unidade o indicador de progresso inferior, o qual irá demonstrar a evolução do processamento de acordo com a quantidade definida pelo método SetRegua2(). Parâmetro: cMensagem

Exemplo: Objeto MsNewProcess() e métodos acessórios

```

/*/
+-----+
| Função   | GPROCES2   | Autor | ROBSON LUIZ   | Data |           |
+-----+
| Descrição | Programa que demonstra a utilização do objeto MsNewProcess()   |
|           | e seus métodos IncReguaX() e SetReguaX() |
|           |
+-----+
| Uso       | Curso ADVPL
|           |
+-----+
/*/

User Function GProces2()
Private oProcess := NIL

oProcess := MsNewProcess():New({|IEnd| RunProc(IEnd,oProcess)};
"Processando","Lendo...",".T.)
oProcess:Activate()

Return Nil

/*/
+-----+
| Função   | RUNPROC   | Autor | ROBSON LUIZ   | Data |           |
+-----+
| Descrição | Função de processamento executada através da MsNewProcess()   |
+-----+
| Uso       |           | Curso ADVPL
|           |
+-----+
/*/

Static Function RunProc(IEnd,oObj)
Local i := 0

```

```
Local aTabela := {}
Local nCnt := 0

aTabela := {"00",0},{"13",0},{"35",0},{"T3",0}}

dbSelectArea("SX5")
cFilialSX5 := xFilial("SX5")
dbSetOrder(1)
For i:=1 To Len(aTabela)
    dbSeek(cFilialSX5+aTabela[i,1])
    While !Eof() .And. X5_FILIAL+X5_TABELA == cFilialSX5+aTabela[i,1]
        If !End
            Exit
        Endif
        nCnt++
        dbSkip()
    End
    aTabela[i,2] := nCnt
    nCnt := 0
Next i

oObj:SetRegua1(Len(aTabela))
For i:=1 To Len(aTabela)
    If !End
        Exit
    Endif
    oObj:IncRegua1("Lendo Tabela: "+aTabela[i,1])
    dbSelectArea("SX5")
    dbSeek(cFilialSX5+aTabela[i,1])
    oObj:SetRegua2(aTabela[i,2])
    While !Eof() .And. X5_FILIAL+X5_TABELA == cFilialSX5+aTabela[i,1]
        oObj:IncRegua2("Lendo chave: "+X5_CHAVE)
        If !End
            Exit
        Endif
        dbSkip()
    End
```

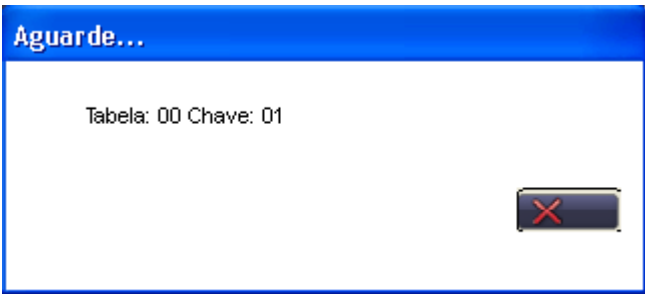

Next i

Return

4.2.4. MsAguarde().

Indicador de processamento sem incremento.

- Sintaxe: Processa(bAcao, cMensagem, cTitulo)
- Retorno: Nil
- Parâmetros:

bAcao	Bloco de código que especifica a ação que será executada com o acompanhamento da régua de processamento.
cMensagem	Mensagem que será exibida na régua de processamento durante a execução.
cTitulo	Título da janela da régua de processamento.
Aparência	

Exemplo: MSAGUARDE()

```

/*/
+-----+
| Função   | GMSAGUARDE | Autor | ROBSON LUIZ   | Data |      |
+-----+
| Descrição | Programa que demonstra a utilização das funções MSAGUARDE() |
|           | e MSPROCTXT() |
|           |
+-----+
| Uso      | Curso ADVPL |
|           |
+-----+

```

```
/*/
```

```
USER FUNCTION GMSAguarde()
```

```
PRIVATE IEnd := .F.
```

```
MsAguarde({|IEnd| RunProc(@IEnd)}, "Aguarde...", "Processando Clientes", .T.)
```

```
RETURN
```

```
/*/
```

```
+-----
```

```
| Função   | RUNPROC   | Autor | ROBSON LUIZ   | Data |      |
```

```
+-----
```

```
| Descrição | Função de processamento          |
```

```
+-----
```

```
| Uso       | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
STATIC FUNCTION RunProc(IEnd)
```

```
dbSelectArea("SX5")
```

```
dbSetOrder(1)
```

```
dbGoTop()
```

```
While !Eof()
```

```
  If IEnd
```

```
    MsgInfo(cCancel, "Fim")
```

```
    Exit
```

```
  Endif
```

```
  MsProcTxt("Tabela: "+SX5->X5_TABELA+" Chave: "+SX5->X5_CHAVE)
```

```
  dbSkip()
```


```
End
```

```
RETURN
```

4.2.5. MsgRun().

Indicador de processamento sem incremento.

- Sintaxe: Processa(cMensagem, cTitulo, bAcao)
- Retorno: Nil
- Parâmetros:

cMensagem	Mensagem que será exibida na régua de processamento durante a execução.
cTitulo	Título da janela da régua de processamento.
bAcao	Bloco de código que especifica a ação que será executada com o acompanhamento da régua de processamento.
Aparência	

Exemplo: MSGRUN()

```

/*/
+-----+
| Função  | GMSGRUN  | Autor | ROBSON LUIZ   | Data |      |
+-----+
| Descrição | Programa que demonstra a utilização das funções MSGRUN()      |
|           | e DBEVAL()                                     |
|           |                                                 |
+-----+
| Uso      | Curso ADVPL
|           |
+-----+
/*/

USER FUNCTION GMsgRun()
LOCAL nCnt := 0

```

```
dbSelectArea("SX1")
```

```
dbGoTop()
```

```
MsgRun("Lendo arquivo, aguarde...", "Título opcional", { | | dbEval({ |x| nCnt++}) })
```

```
MsgInfo("Ufa!!!, li "+AllTrim(Str(nCnt))+ " registros", FunName())
```

```
RETURN
```

4.3. ListBox()

A sintaxe clássica da linguagem ADVPL permite que o componente visual ListBox implemente dois tipos distintos de objetos:

- Lista simples: lista de apenas uma coluna no formato de um vetor, a qual não necessita da especificação de um cabeçalho.
- Lista com colunas: lista com diversas colunas que necessita de um cabeçalho no formato de um aHeader (array de cabeçalho).

4.3.1. ListBox simples

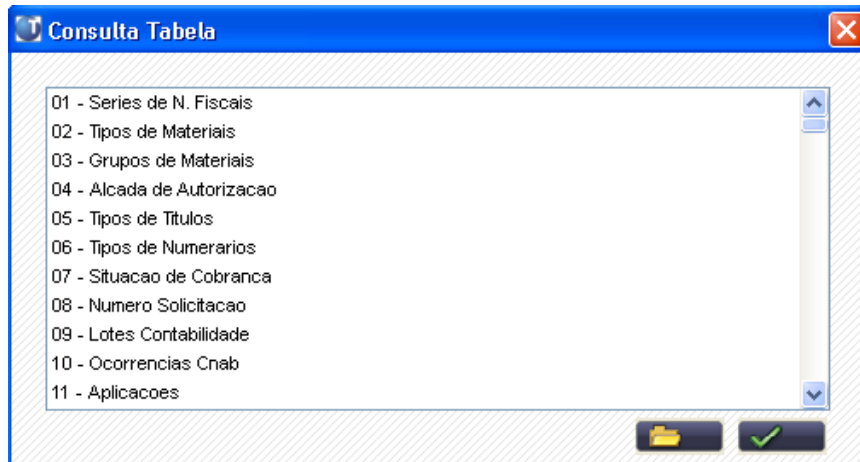
- Sintaxe:

@ nLinha,nColuna LISTBOX oListBox VAR nLista ITEMS aLista SIZE nLargura,nAltura OF oObjetoRef UNIDADE ON CHANGE CHANGE

- Parâmetros:

nLinha,nColuna	Posição do objeto ListBox em função da janela em que ele será definido.
oListBox	Objeto ListBox que será criado.
nLista	Variável numérica que contém o número do item selecionado no ListBox.
aLista	Vetor simples contendo as strings que serão exibidas no ListBox.
nLargura,nAltura	Dimensões do objeto ListBox.
oObjetoRef	Objeto dialog no qual o componente será definido.
UNIDADE	Unidade de medida das dimensões: PIXEL.
CHANGE	Função ou lista de expressões que será executada na seleção de um item do ListBox.

- Aparência:



Exemplo: LISTBOX como lista simples

```
#include "protheus.ch"
```

```
/*/
```

```
+-----+
| Função   | LISTBOXITE | Autor | ROBSON LUIZ   | Data |      |
+-----+
| Descrição | Programa que demonstra a utilização do LISTBOX() como lista |
|                                     | simples.
|                                     |
+-----+
| Uso      | Curso ADVPL
|          |
+-----+
```

```
/*/
```

```
User Function ListBoxIte()
```

```
Local aVetor := {}
```

```
Local oDlg   := Nil
```

```
Local oLbx   := Nil
```

```
Local cTitulo := "Consulta Tabela"
```

```
Local nChave  := 0
```

```
Local cChave  := ""
```

```
dbSelectArea("SX5")
dbSetOrder(1)
dbSeek(xFilial("SX5"))

CursorWait()

//+-----+
//| Carrega o vetor conforme a condição |
//+-----+
While !Eof() .And. X5_FILIAL == xFilial("SX5") .And. X5_TABELA=="00"
    AADD( aVetor, Trim(X5_CHAVE)+" - "+Capital(Trim(X5_DESCRI)) )
    dbSkip()
End

CursorArrow()

If Len( aVetor ) == 0
    Aviso( cTitulo, "Não existe dados a consultar", {"Ok"} )
    Return
Endif

//+-----+
//| Monta a tela para usuário visualizar consulta |
//+-----+
DEFINE MSDIALOG oDlg TITLE cTitulo FROM 0,0 TO 240,500 PIXEL
@ 10,10 LISTBOX oLbx VAR nChave ITEMS aVetor SIZE 230,95 OF oDlg PIXEL
oLbx:bChange := { | | cChave := SubStr(aVetor[nChave],1,2) }
DEFINE SBUTTON FROM 107,183 TYPE 14 ACTION LoadTable(cChave) ENABLE OF oDlg
DEFINE SBUTTON FROM 107,213 TYPE 1 ACTION oDlg:End() ENABLE OF oDlg

ACTIVATE MSDIALOG oDlg CENTER

Return
```

Exemplo: LISTBOX como lista simples – funções acessórias

```

/*/
+-----+
| Função | LISTBOXITE | Autor | ROBSON LUIZ | Data | |
+-----+
| Descrição | Função que carrega os dados da tabela selecionada em um |
| | | listbox. |
|
+-----+
| Uso | Curso ADVPL |
|
+-----+
/*/

STATIC FUNCTION LoadTable(cTabela)

LOCAL aTabela := {}
LOCAL oDlg := NIL
LOCAL oLbx := NIL

dbSelectArea("SX5")
dbSeek(xFilial("SX5")+cTabela)

//+-----+
//| O vetor pode receber carga de duas maneiras, acompanhe... |
//+-----+
//| Utilizando While/End |
//+-----+

dbEval({ | | AADD(aTabela,{X5_CHAVE,Capital(X5_DESCR)})},,{ | | X5_TABELA==cTabela})

If Len(aTabela)==0
    Aviso( "FIM", "Necessário selecionar um item", {"Ok"} )
    Return
Endif

```

```

DEFINE MSDIALOG oDlg TITLE "Dados da tabela selecionada" FROM 300,400 TO 540,900 PIXEL
@ 10,10 LISTBOX oLbx FIELDS HEADER "Tabela", "Descrição" SIZE 230,095 OF oDlg PIXEL
oLbx:SetArray( aTabela )
oLbx:bLine := { | | {aTabela[oLbx:nAt,1],aTabela[oLbx:nAt,2]} }
DEFINE SBUTTON FROM 107,213 TYPE 1 ACTION oDlg:End() ENABLE OF oDlg
ACTIVATE MSDIALOG oDlg

```

RETURN

4.3.2. ListBox múltiplas colunas

- Sintaxe:

@ nLinha,nColuna LISTBOX oListBox FIELDS HEADER "Header1", ..., "HeaderX" SIZE nLargura,nAltura OF oObjetoRef UNIDADE

- Parâmetros:

nLinha,nColuna	Posição do objeto ListBox em função da janela em que ele será definido.
oListBox	Objeto ListBox que será criado.
nLista	Variável numérica que contém o número do item selecionado no ListBox.
"Header1",...,"HeaderX"	Strings identificando os títulos das colunas do Grid.
nLargura,nAltura	Dimensões do objeto ListBox.
oObjetoRef	Objeto dialog no qual o componente será definido.
UNIDADE	Unidade de medida das dimensões: PIXEL.
CHANGE	Função ou lista de expressões que será executada na seleção de um item do ListBox.

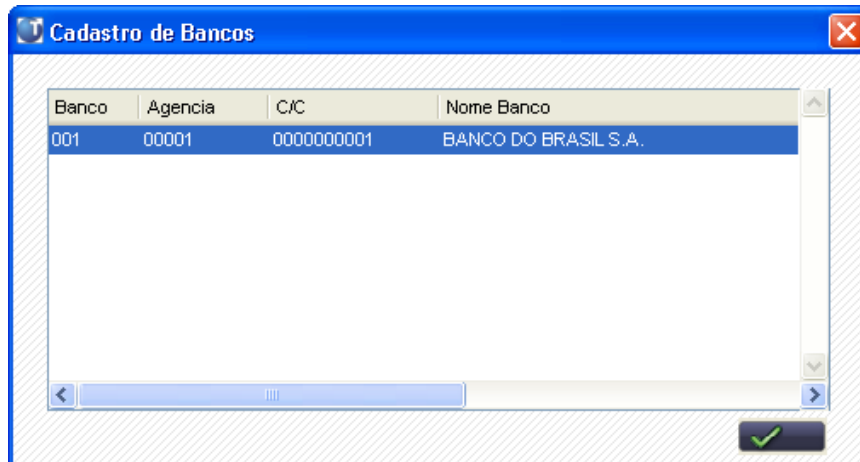
- Métodos:

SetArray()	Método o objeto ListBox que define qual o array contém os dados que serão exibidos no grid.
-------------------	---

- Atributos:

bLine	Atributo do objeto ListBox que vincula cada linha,coluna do array, com cada cabeçalho do grid.
--------------	--

- Aparência:



Exemplo: LISTBOX com grid

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função | LIST_BOX | Autor | ROBSON LUIZ | Data | |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização de um  
LISTBOX() com |  
| | grid.
```

```
| |
```

```
+-----
```

```
| Uso | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
User Function List_Box()
```

```
Local aVetor := {}
```

```
Local oDlg
```

```
Local oLbx
```

```
Local cTitulo := "Cadastro de Bancos"
```

```
Local cFilSA6
```

```
dbSelectArea("SA6")
dbSetOrder(1)
cFilSA6 := xFilial("SA6")
dbSeek(cFilSA6)

// Carrega o vetor conforme a condição.
While !Eof() .And. A6_FILIAL == cFilSA6
    AADD( aVetor, { A6_COD, A6_AGENCIA, A6_NUMCON, A6_NOME, A6_NREDUZ, A6_BAIRRO,
A6_MUN } )
        dbSkip()
End

// Se não houver dados no vetor, avisar usuário e abandonar rotina.
If Len( aVetor ) == 0
    Aviso( cTitulo, "Não existe dados a consultar", {"Ok"} )
    Return
Endif

// Monta a tela para usuário visualizar consulta.
DEFINE MSDIALOG oDlg TITLE cTitulo FROM 0,0 TO 240,500 PIXEL

// Primeira opção para montar o listbox.
@ 10,10 LISTBOX oLbx FIELDS HEADER ;
"Banco", "Agencia", "C/C", "Nome Banco", "Fantasia", "Bairro", "Município" ;
SIZE 230,95 OF oDlg PIXEL

oLbx:SetArray( aVetor )
oLbx:bLine := { | | {aVetor[oLbx:nAt,1],;
    aVetor[oLbx:nAt,2],;
    aVetor[oLbx:nAt,3],;
    aVetor[oLbx:nAt,4],;
    aVetor[oLbx:nAt,5],;
    aVetor[oLbx:nAt,6],;
    aVetor[oLbx:nAt,7]}}

// Segunda opção para monta o listbox
/*
```

```
oLbx := TWBrowse():New(10,10,230,95,,aCabecalho,,oDlg,,,,,,,,,F,,.T,,.F,,)
oLbx:SetArray( aVetor )
oLbx:bLine := { | | aEval(aVetor[oLbx:nAt],{|z,w| aVetor[oLbx:nAt,w] } ) }
*/
```

```
DEFINE SBUTTON FROM 107,213 TYPE 1 ACTION oDlg:End() ENABLE OF oDlg
ACTIVATE MSDIALOG oDlg CENTER
```

Return

4.4. ScrollBox()

O ScrollBox é o objeto utilizado para permitir a um Dialog exibir barras de rolagem verticais e Horizontais. Algumas aplicações com objetos definem automaticamente o ScrollBox, tais como:

- Enchoice() ou MsMGet()
- NewGetDados()
- ListBox()

Quando é definido um objeto ScrollBox, os demais componentes da janela deverão referenciar este objeto e não mais o objeto Dialog.

Desta forma o ScrollBox é atribuído a um objeto Dialog, e os componentes ao ScrollBox.

- MsDialog() ß ScrollBox()
- ScrollBox() ß Componentes Visuais
- Sintaxe:

@ nLinha,nColuna SCROLLBOX oScrollBox HORIZONTAL VERTICAL SIZE nLargura,nAltura OF oObjetoRef BORDER

- Parâmetros:

nLinha,nColuna	Posição do objeto ScrollBox em função da janela em que ele será definido.
oScrollBox	Objeto ScrollBox que será criado.
HORIZONTAL	Parâmetro que quando definido habilita a régua de rolagem horizontal.
VERTICAL	Parâmetro que quando definido habilita a régua de rolagem vertical.
nLargura,nAltura	Dimensões do objeto ScrollBox.
oObjetoRef	Objeto dialog no qual o componente será definido.

BORDER	Parâmetro que quando definido habilita a exibição de uma borda de delimitação do ScrollBox em relação a outros objetos.
---------------	---

Exemplo: Utilização de múltiplos ScrollBoxes

```
#INCLUDE "PROTHEUS.CH"
```

```
/*/
```

```
+-----
```

```
| Função   | SCROLL() | Autor | ROBSON LUIZ | Data |      |
```

```
+-----
```

```
| Descrição | Programa que demonstra como montar uma enchoice apenas |
```

```
|                                     | com variáveis, incluindo o recurso de rolagem. |
```

```
+-----
```

```
| Uso       | Curso ADVPL                                     |
```

```
+-----
```

```
/*/
```

```
USER FUNCTION Scroll()
```

```
LOCAL oDlg := NIL
```

```
LOCAL oScroll := NIL
```

```
LOCAL oLbx1 := NIL
```

```
LOCAL oLbx2 := NIL
```

```
LOCAL bGet := NIL
```

```
LOCAL oGet := NIL
```

```
LOCAL aAIIPM := {}
```

```
LOCAL aTitulo := {}
```

```
LOCAL nTop := 5
```

```
LOCAL nWidth := 0
```

```
LOCAL cGet := ""
```

```
LOCAL cPict := ""
```

```
LOCAL cVar := ""
```

```
LOCAL n := 0
```

```
PRIVATE cTitulo := "Consulta Parcelamento"
```

```
PRIVATE aSay := {}
PRIVATE cProcesso,cPrefixo,cTipo,cCliente,cLoja,cNome,cCGC
PRIVATE dData,nTotal,nUFESP,cStatus,cCond
```

```
cProcesso := "P00001"
cPrefixo := "UNI"
cTipo := "MAN"
cCliente := "000001"
cLoja := "01"
cNome := "JOSE DA SILVA SANTOS SOARES"
cCGC := "00.000.000/0001-91"
dData := "26/03/03"
nTotal := 5922.00
nUFESP := 1000.00
cStatus := "Z"
cCond := "001"
```

```
// Vetor para os campos no Scrooll Box
//+-----+
//| aSay[n][1] - Titulo |
//| aSay[n][2] - Tipo |
//| aSay[n][3] - Tamanho |
//| aSay[n][4] - Decimal |
//| aSay[n][5] - Conteúdo/Variável |
//| aSay[n][6] - Formato |
//+-----+
AADD(aSay,{"Processo" ,"C",06,0,"cProcesso" ,"@"})
AADD(aSay,{"Prefixo" ,"C",03,0,"cPrefixo" ,"@"})
AADD(aSay,{"Tipo" ,"C",03,0,"cTipo" ,"@"})
AADD(aSay,{"Cliente" ,"C",06,0,"cCliente" ,"@"})
AADD(aSay,{"Loja" ,"C",02,0,"cLoja" ,"@"})
AADD(aSay,{"Nome" ,"C",30,0,"cNome" ,"@"})
AADD(aSay,{"CNPJ/CPF" ,"C",14,0,"cCGC" ,"@"})
AADD(aSay,{"Dt.Processo" ,"D",08,0,"dData" ,"@"})
AADD(aSay,{"Total R$" ,"N",17,2,"nTotal" ,"@"})
AADD(aSay,{"Total UFESP" ,"N",17,2,"nUFESP" ,"@"})
AADD(aSay,{"Status" ,"C",01,0,"cStatus" ,"@"})
```

```
AADD(aSay,{"Cond.Pagto" ,"C",03,0,"cCond"  ,"@!"})
```

```
// Vetor para List Box
```

```
AADD(aAIIPM,{"1234","DCD9815","26/03/03"})
```

```
AADD(aAIIPM,{"1234","DCD9815","26/03/03"})
```

```
AADD(aAIIPM,{"1234","DCD9815","26/03/03"})
```

```
// Vetor para List Box
```

```
AADD(aTitulo,{"A","26/03/03","26/03/03","1.974,00","100,00"})
```

```
AADD(aTitulo,{"A","26/03/03","26/03/03","1.974,00","100,00"})
```

```
AADD(aTitulo,{"A","26/03/03","26/03/03","1.974,00","100,00"})
```

```
DEFINE MSDIALOG oDlg TITLE cTitulo FROM 122,0 TO 432,600 OF oDlg PIXEL
```

```
@ 013,002 TO 154,192 LABEL "Parcelamento" OF oDlg PIXEL
```

```
@ 013,195 TO 082,298 LABEL "Títulos" OF oDlg PIXEL
```

```
@ 083,195 TO 154,298 LABEL "AIIPM" OF oDlg PIXEL
```

```
//scrollbox
```

```
@ 019,006 SCROLLBOX oScroll HORIZONTAL VERTICAL SIZE 131,182 OF oDlg BORDER
```

```
For n:=1 TO Len(aSay)
```

```
  bGet := &("{ | | '"+aSay[n][1]+'"}")
```

```
  cVar := aSay[n][5]
```

```
  cGet := "{ | u | IIF(PCount()>0,"+cVar+":u,"+cVar+")}"
```

```
  cPict := aSay[n][6]
```

```
  TSay():New(nTop,5,bGet,oScroll,,,,.F.,.F.,.F.,.T.,,,;
```

```
  GetTextWidth(0,Trim(aSay[n][1])),15,;
```

```
  .F.,.F.,.F.,.F.,.F.)
```

```
  oGet:=TGet():New(nTop-2,40,&cGet,oScroll,,7,cPict,,,,.F.,.T.,;
```

```
  .F.,.F.,.F.,.T.,.F.,(cVar),,,.T.)
```

```
  nTop+=11
```

```
Next n
```

```
//listbox títulos
```

```
@ 019,199 LISTBOX oLbx1 FIELDS HEADER ;
```

```
"Parcela","Vencto","Vencto.Real","Valor R$","Qtd.UFESP";
```

```
COLSIZES 21,24,33,63,100;
SIZE 095,059 OF oDlg PIXEL
oLbx1:SetArray( aTitulo )
oLbx1:bLine := { | {aTitulo[oLbx1:nAt,1],aTitulo[oLbx1:nAt,2],;
aTitulo[oLbx1:nAt,3],aTitulo[oLbx1:nAt,4],aTitulo[oLbx1:nAt,5]}}

//listbox aiipm
@ 089,199 LISTBOX oLbx2 FIELDS HEADER "AIIPM","Placa","Data Multa" ;
COLSIZES 24,21,30 SIZE 095,061 OF oDlg PIXEL
oLbx2:SetArray( aAIIPM )
oLbx2:bLine := { | {aAIIPM[oLbx2:nAt,1],aAIIPM[oLbx2:nAt,2],aAIIPM[oLbx2:nAt,3]}}

ACTIVATE MSDIALOG oDlg CENTER ON INIT EnchoiceBar(oDlg,{ | oDlg:End()},{ | oDlg:End()})

RETURN
```


4.5. ParamBox()

Implementa uma tela de parâmetros, que não necessita da criação de um grupo de perguntas no SX1, e com funcionalidades que a Pergunte() não disponibiliza, tais como CheckBox e RadioButtons.

Cada componente da ParamBox será associado a um parâmetro Private denominado MV_PARxx, de acordo com a ordem do componente na tela. Os parâmetros da ParamBox podem ser utilizados de forma independente em uma rotina específica, ou complementando opções de uma rotina padrão.

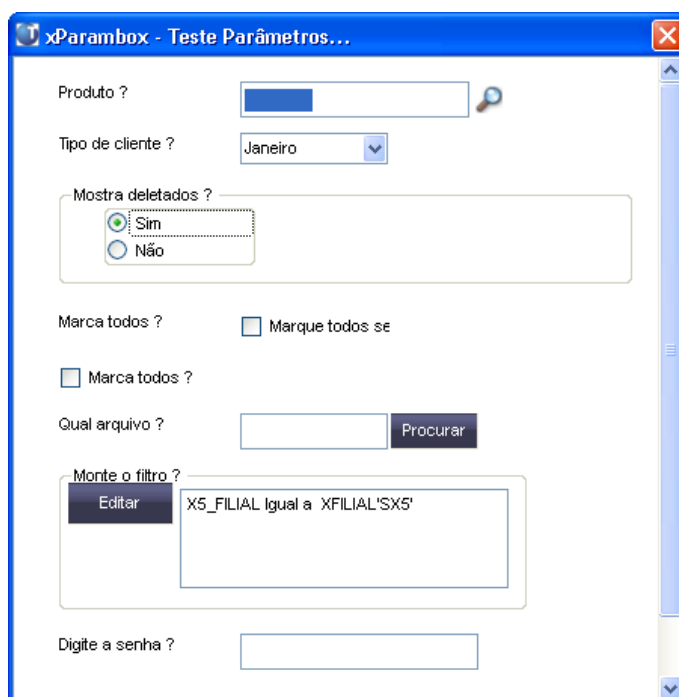
Cuidados

- A PARAMBOX define os parâmetros seguindo o princípio das variáveis MV_PARxx. Caso ela seja utilizada em uma rotina em conjunto com parâmetros padrões (SX1 + Pergunte()) é necessário salvar os parâmetros padrões, chamar a Parambox(), salvar o retorno da Parambox() em variáveis Private específicas (MVPARBOXxx) e depois restaurar os parâmetros padrões, conforme o exemplo desta documentação.
- O objeto COMBO() da PARAMBOX() possui um problema em seu retorno: Caso o combo não seja selecionado, ele manterá seu conteúdo como numérico, caso seja ele receberá o texto da opção e não o número da opção. O exemplo desta documentação ilustra o tratamento de código necessário para proteger a aplicação.
- Ao utilizar a ParamBox em uma função que também utilize parâmetros definidos pela função Pergunte() deve-se:

- Salvar e restaurar os MV_PARs da Pergunte()
- Definir variáveis Private próprias para a ParamBox, as quais irão armazenar o conteúdo das MV_PARs que esta retorna.
- Sintaxe: ParamBox (aParamBox, cTitulo, aRet, bOk, aButtons, lCentered,, nPosx, nPosy, oMainDlg, cLoad, lCanSave, lUserSave)
- Retorno: IOK  indica se a tela de parâmetros foi cancelada ou confirmada
- Parâmetros:

aParamBox	Array de parâmetros de acordo com a regra da ParamBox
cTitulo	Titulo da janela de parâmetros
aRet	Array que será passado por referencia e retornado com o conteúdo de cada parâmetro
bOk	Bloco de código para validação do OK da tela de parâmetros
aButtons	Array contendo a regra para adição de novos botões (além do OK e Cancelar) // AADD(aButtons,{nType,bAction,cTexto})
lCentered	Se a tela será exibida centralizada, quando a mesma não estiver vinculada a outra janela
nPosx	Posição inicial -> linha (Linha final: nPosX+274)
nPosy	Posição inicial -> coluna (Coluna final: nPosY+445)
oMainDlg	Caso o ParamBox deva ser vinculado a uma outra tela
cLoad	Nome do arquivo aonde as respostas do usuário serão salvas / lidas
lCanSave	Se as respostas para as perguntas podem ser salvas
lUserSave	Se o usuário pode salvar sua própria configuração.

- Aparência:



- Regras do array aParamBox:

[1] Tipo do parâmetro: Para cada tipo de parâmetro as demais posições do array variam de conteúdo conforme abaixo:

1 - MsGet

- [2] : Descrição
- [3] : String contendo o inicializador do campo
- [4] : String contendo a Picture do campo
- [5] : String contendo a validação
- [6] : Consulta F3
- [7] : String contendo a validação When
- [8] : Tamanho do MsGet
- [9] : Flag .T./.F. Parâmetro Obrigatório ?

2 - Combo

- [2] : Descrição
- [3] : Numérico contendo a opção inicial do combo
- [4] : Array contendo as opções do Combo
- [5] : Tamanho do Combo
- [6] : Validação
- [7] : Flag .T./.F. Parâmetro Obrigatório ?

3 - Radio

- [2] : Descrição
- [3] : Numérico contendo a opção inicial do Radio
- [4] : Array contendo as opções do Radio
- [5] : Tamanho do Radio
- [6] : Validação
- [7] : Flag .T./.F. Parâmetro Obrigatório ?

4 - CheckBox (Com Say)

- [2] : Descrição
- [3] : Indicador Lógico contendo o inicial do Check
- [4] : Texto do CheckBox
- [5] : Tamanho do Radio
- [6] : Validação
- [7] : Flag .T./.F. Parâmetro Obrigatório ?

5 - CheckBox (linha inteira)

- [2] : Descrição
- [3] : Indicador Lógico contendo o inicial do Check
- [4] : Tamanho do Radio
- [5] : Validação
- [6] : Flag .T./.F. Parâmetro Obrigatório ?

6 - File

- [2] : Descrição
- [3] : String contendo o inicializador do campo
- [4] : String contendo a Picture do campo
- [5] : String contendo a validação
- [6] : String contendo a validação When
- [7] : Tamanho do MsGet
- [8] : Flag .T./.F. Parâmetro Obrigatório ?
- [9] : Texto contendo os tipos de arquivo
Ex.: "Arquivos .CSV | *.CSV"
- [10]: Diretório inicial do CGETFILE()
- [11]: Parâmetros do CGETFILE()

7 - Montagem de expressão de filtro

- [2] : Descrição
- [3] : Alias da tabela
- [4] : Filtro inicial
- [5] : Opcional - Clausula When Botão Editar Filtro

8 - MsGet Password

- [2] : Descrição
- [3] : String contendo o inicializador do campo
- [4] : String contendo a Picture do campo
- [5] : String contendo a validação
- [6] : Consulta F3
- [7] : String contendo a validação When
- [8] : Tamanho do MsGet
- [9] : Flag .T./.F. Parâmetro Obrigatório ?

9 - MsGet Say

[2] : String Contendo o Texto a ser apresentado

[3] : Tamanho da String

[4] : Altura da String

[5] : Negrito (lógico)

Exemplo: Utilização da ParamBox()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função   | xParamBox | Autor | ROBSON LUIZ   | Data |   |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização da PARAMBOX como |
```

```
|                                     | forma alternativa de disponibilizar parâmetros
```

```
em um |
```

```
|      | processamento.                                     |
```

```
+-----
```

```
| Uso      | Curso ADVPL                                     |
```

```
+-----
```

```
/*/
```

```
User Function xParamBox()
```

```
Local aRet := {}
```

```
Local aParamBox := {}
```

```
Local                                     aCombo                                     :=
```

```
{"Janeiro","Fevereiro","Março","Abril","Maio","Junho","Julho","Agosto","Setembro","Outubro",  
,"Novembro","Dezembro"}
```

```
Local i := 0
```

```
Private cCadastro := "xParambox"
```

```
AADD(aParamBox,{1,"Produto",Space(15),"","SB1","",0,.F.})
```

```
AADD(aParamBox,{2,"Tipo de cliente",1,aCombo,50,"",.F.})
```

```
AADD(aParamBox,{3,"Mostra deletados",IIF(Set(_SET_DELETED),1,2),{"Sim","Não"},50,"",.F.})
```

```
AADD(aParamBox,{4,"Marca todos ?",".F.,"Marque todos se necessário for.",50,"",".F.})
```

```
AADD(aParamBox,{5,"Marca todos ?",".F.,50,"",".F.})
```

```
AADD(aParamBox,{6,"Qual arquivo",Space(50),"","","",50,.F.,;
"Arquivo .DBF | *.DBF"})
```

```
AADD(aParamBox,{7,"Monte o filtro","SX5","X5_FILIAL==xFilial('SX5')"})
```

```
AADD(aParamBox,{8,"Digite a senha",Space(15),"","","","",80,.F.})
```

```
If ParamBox(aParamBox,"Teste Parâmetros...",&aRet)
```

```
  For i:=1 To Len(aRet)
```

```
    MsgInfo(aRet[i],"Opção escolhida")
```

```
  Next
```

```
Endif
```

```
Return
```

Exemplo: Protegendo os parâmetros MV_PARs da Pergunte() em uso.

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função   | XPARBOX() | Autor | ARNALDO RAYMUNDO JR. | Data | |
```

```
+-----
```

```
| Descrição | Função utilizando a PARAMBOX() e protegendo os MV_PARs |
```

```
| | | ativos do programa principal.
```

```
|
```

```
+-----
```

```
| Uso      | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
Static Function XPARGBOX(cPerg)
```

```
Local aParamBox := {}
```

```

Local cTitulo                := "Transferência para Operação"
Local bOk                    := {| | .T.}
Local aButtons               := {}; Local aRet := {}
Local nPosx; Local nPosy; Local nX          := 0
Local cLoad                  := ""
Local lCentered              := .T.; Local lCanSave := .F.; Local lUserSave := .F.
Local aParamAtu := Array(4)

// Salva as perguntas padrões antes da chamada da ParamBox
For nX := 1 to Len(aParamAtu)
    aParamAtu [nX]            := &("Mv_Par"+StrZero(nX,2))
Next nX

AADD(aParamBox,{2,"Atualiza taxa de depreciação?", 2, {"Sim","Não"}, 100,;
    "AlwaysTrue()".T.})

ParamBox(aParamBox, cTitulo, aRet, bOk, aButtons, lCentered, nPosx, nPosy, /*oMainDlg*/ ,;
    cLoad, lCanSave, lUserSave)

IF ValType(aRet) == "A" .AND. Len(aRet) == Len(aParamBox)
    For nX := 1 to Len(aParamBox)
        If aParamBox[nX][1] == 1
            &("MvParBox"+StrZero(nX,2)) := aRet[nX]
        Elseif aParamBox[nX][1] == 2 .AND.
            ValType(aRet[nX]) == "C"
                &("MvParBox"+StrZero(nX,2)) :=
aScan(aParamBox[nX][4],;
    {|x| Alltrim(x) == aRet[nX]})
        Elseif aParamBox[nX][1] == 2 .AND.
            ValType(aRet[nX]) == "N"
                &("MvParBox"+StrZero(nX,2)) := aRet[nX]
        Endif
    Next nX
ENDIF

// Restaura as perguntas padrões apos a chamada da ParamBox
For nX := 1 to Len(aParamAtu)

```

```
&("Mv_Par"+StrZero(nX,2)) := aParamAtu[nX]
```

Next nX

Return

MÓDULO 06: ADVPL Orientado à objetos I

Neste módulo serão tratados os componentes e objetos da interface visual da linguagem ADVPL, permitindo o desenvolvimento de aplicações com interfaces gráficas com sintaxe orientada a objetos.

5. Componentes da interface visual do ADVPL

A linguagem ADVPL possui diversos componentes visuais e auxiliares, os quais podem ser representados utilizando a estrutura abaixo:

- Classes da Interface Visual
 - Tsrvobject
 - Classes Auxiliares
 - Tfont
 - Classes de Janelas
 - Msdialog
 - Tdialog
 - Twindow
 - Classes de Componentes
 - Tcontrol
 - Classes de ComponentesVisuais
 - Brgetddb
 - Mscalend
 - Mscalendgrid
 - Msselbr
 - Msworktime
 - Sbutton
 - Tbar
 - Tbitmap
 - Tbrowsebutton
 - Tbtnbmp
 - Tbtnbmp2
 - Tbutton
 - Tcbrowse
 - Tcheckbox
 - Tcolortriangle
 - Tcombobox
 - Tfolder
 - Tfont

Tget
Tgroup
Thbutton
Tibrowser
Tlistbox
Tmenu
Tmenubar
Tmeter
Tmsgraphic
Tmsgbar
Tmultibtn
Tmultiget
Tolecontainer
Tpageview
Tpanel
Tradmenu
Tsbrowse
Tsay
Tscrollbox
Tsimpleeditor
Tslider

- Classes de Componentes Visuais

Tsplitter
Ttabs
Ttoolbox
Twbrowse
Vcbrowse

Classes da interface visual

TSRVOBJECT()

Descrição	Classe abstrata inicial de todas as classes de interface do ADVPL. Não deve ser instanciada diretamente.
------------------	--

Classes auxiliares

TFONT()

Descrição	Classe de objetos que define a fonte do texto utilizado nos controles visuais.
------------------	--

Classes de janelas**MSDIALOG()**

Descrição	Classe de objetos que deve ser utilizada como padrão de janela para entrada de dados. MSDialog é um tipo de janela diálogo modal, isto é, não permite que outra janela ativa receba dados enquanto esta estiver ativa.
------------------	--

TDIALOG()

Descrição	Classe de objetos do tipo diálogo de entrada de dados, sendo seu uso reservado. Recomenda-se utilizar a classe MSDialog que é herdada desta classe.
------------------	---

TWINDOW()

Descrição	Classe de objetos do tipo diálogo principal de programa. Deverá existir apenas uma instância deste objeto na execução do programa.
------------------	--

Classes de componentes**TCONTROL()**

Descrição	Classe abstrata comum entre todos os componentes visuais editáveis. Não deve ser instanciada diretamente.
------------------	---

Classes de componentes visuais**BRGETDDB()**

Descrição	Classe de objetos visuais do tipo Grid.
------------------	---

MSCALEND()

Descrição	Classe de objetos visuais do tipo Calendário.
------------------	---

MSCALENDGRID()

Descrição	Classe de objetos visuais do tipo Grade de Períodos.
------------------	--

MSSELBR()

Descrição	Classe de objetos visuais do tipo controle - Grid
------------------	---

MSWORKTIME()

Descrição	Classe de objetos visuais do tipo controle - Barra de Período.
------------------	--

SBUTTON()

Descrição	Classe de objetos visuais do tipo botão, o qual pode possuir imagens pad associadas ao seu tipo.
------------------	--

TBAR()

Descrição	Classe de objetos visuais do tipo Barra Superior.
------------------	---

TBITMAP()

Descrição	Classe de objetos visuais que permite a exibição de uma imagem.
------------------	---

TBROWSEBUTTON()

Descrição	Classe de objetos visuais do tipo botão no formato padrão utilizado browses da aplicação.
------------------	---

TBTNBMP()

Descrição	Classe de objetos visuais do tipo botão, o qual permite que seja vinculada uma imagem ao controle.
------------------	--

TBTNBMP2()

Descrição	Classe de objetos visuais do tipo botão, o qual permite a exibição de imagem ou de um popup.
------------------	--

TBUTTON()

Descrição	Classe de objetos visuais do tipo botão, o qual permite a utilização de t para sua identificação.
------------------	---

TCBROWSE()

Descrição	Classe de objetos visuais do tipo Grid.
------------------	---

TCHECKBOX()

Descrição	Classe de objetos visuais do tipo controle - CheckBox.
------------------	--

TCOLORTRIANGLE()

Descrição	Classe de objetos visuais do tipo Paleta de Cores.
------------------	--

TCOMBOBOX()

Descrição	Classe de objetos visuais do tipo tComboBox, a qual cria uma entrada de dados com múltipla escolha com item definido em uma lista vertical, acionada pelo botão direito ou pelo botão esquerdo localizado na parte direita do controle. A variável associada ao controle terá o valor de um dos itens selecionados ou no caso de uma lista indexada, o valor de seu índice.
------------------	---

TFOLDER()

Descrição	Classe de objetos visuais do tipo controle - Folder.
------------------	--

TGET()

Descrição	Classe de objetos visuais do tipo controle – tGet, a qual cria um controle que armazena ou altera o conteúdo de uma variável através de digitação. O conteúdo da variável só é modificado quando o controle perde o foco de edição para outro controle.
------------------	---

TGROUP()

Descrição	Classe de objetos visuais do tipo painel – tGroup, a qual cria um painel onde controles visuais podem ser agrupados ou classificados. Neste painel é criada uma borda com título em volta dos controles agrupados.
------------------	--

THBUTTON()

Descrição	Classe de objetos visuais do tipo botão com hiperlink.
------------------	--

TIBROWSER()

Descrição	Classe de objetos visuais do tipo Página de Internet, sendo necessário incluir a cláusula BrowserEnabled=1 no Config do Remote.INI
------------------	--

TLISTBOX()

Descrição	Classe de objetos visuais do tipo controle – tListbox, a qual cria uma janela com itens selecionáveis e barra de rolagem. Ao selecionar um item, uma variável é atualizada com o conteúdo do item selecionado.
------------------	--

TMENU()

Descrição	Classe de objetos visuais do tipo controle - Menu.
------------------	--

TMENUBAR()

Descrição	Classe de objetos visuais do tipo controle - Barra de Menu.
------------------	---

TMETER()

Descrição	Classe de objetos visuais do tipo controle – tMeter, a qual exibe uma régua (gauge) de processamento, descrevendo o andamento de um processo através da exibição de uma barra horizontal.
------------------	---

TMSGGRAPHIC()

Descrição	Classe de objetos visuais do tipo controle - Gráfico.
------------------	---

TMSGBAR()

Descrição	Classe de objetos visuais do tipo controle - Rodapé.
------------------	--

TMULTIBTN()

Descrição	Classe de objetos visuais do tipo controle - Múltiplos botões.
------------------	--

TMULTIGET()

Descrição	Classe de objetos visuais do tipo controle - edição de texto de múltiplas linhas.
------------------	---

TOLECONTAINER()

Descrição	Classe de objetos visuais do tipo controle, a qual permite a criação de um botão vinculado a um objeto OLE.
------------------	---

TPAGEVIEW()

Descrição	Classe de objetos visuais do tipo controle, que permite a visualização de arquivos no formato gerado pelo spool de impressão do Protheus.
------------------	---

TPANEL()

Descrição	Classe de objetos visuais do tipo controle – tPanel, a qual permite criar um painel estático, onde podem ser criados outros controles com o objetivo de organizar ou agrupar componentes visuais.
------------------	---

TRADMENU()

Descrição	Classe de objetos visuais do tipo controle – TRadMenu, a qual permite criar um controle visual no formato Radio Button.
------------------	---

TSBROWSE()

Descrição	Classe de objetos visuais do tipo controle – TSBrowse, a qual permite criar um controle visual do tipo Grid.
------------------	--

TSAY()

Descrição	Classe de objetos visuais do tipo controle – tSay, a qual exibe o conteúdo de texto estático sobre uma janela ou controle previamente definidos.
------------------	--

TSCROLLBOX()

Descrição	Classe de objetos visuais do tipo controle – tScrollbar, a qual permite criar um painel com scroll deslizantes nas laterais (horizontais e verticais) do controle.
------------------	--

TSIMPLEEDITOR()

Descrição	Classe de objetos visuais do tipo controle – tSimpleEditor, a qual permite criar um controle visual para edição de textos com recursos simples, como o NotePad®
------------------	---

TSLIDER()

Descrição	Classe de objetos visuais do tipo controle – tSlider, a qual permite criar um controle visual do tipo botão deslizante.
------------------	---

TSPLITTER()

Descrição	Classe de objetos visuais do tipo controle – tSplitter, a qual permite criar um controle visual do tipo divisor.
------------------	--

TTABS()

Descrição	Classe de objetos visuais do tipo controle – TTabs, a qual permite criar um controle visual do tipo pasta.
------------------	--

TTOOLBOX()

Descrição	Classe de objetos visuais do tipo controle – tToolbox, a qual permite criar um controle visual para agrupar diferentes objetos.
------------------	---

TWBROWSE()

Descrição	Classe de objetos visuais do tipo controle – TWBrowse, a qual permite criar um controle visual do tipo Grid.
------------------	--

VCBROWSE()

Descrição	Classe de objetos visuais do tipo controle – VCBrowse, a qual permite criar um controle visual do tipo Grid.
------------------	--

Documentação dos componentes da interface visual

Os componentes da interface visual da linguagem ADVPL utilizados neste treinamento estão documentados na seção Guia de Referência, ao final deste material.

Para visualizar a documentação completa de todos os componentes mencionados neste capítulo deve ser acesso o site TDN – TOTVS DEVELOPER NETWORK (tdn.totvs.com.br).

5.1. Particularidades dos componentes visuais**5.1.1. Configurando as cores para os componentes**

Os componentes visuais da linguagem ADVPL utilizam o padrão de cores RGB.

As cores deste padrão são definidas pela seguinte fórmula, a qual deve ser avaliada tendo como base a paleta de cores no formato RGB:

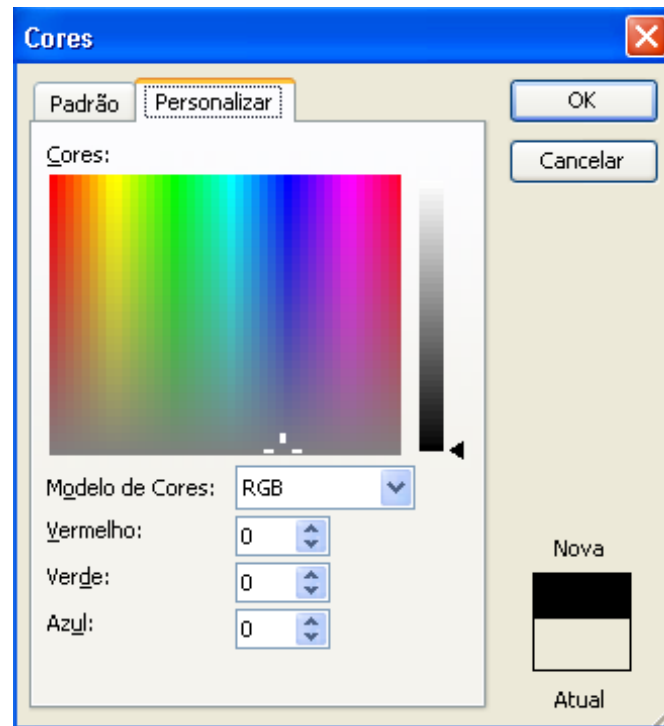
$$nCor := nVermelho + (nVerde * 256) + (nAzul * 65536)$$


Figura: Paleta de cores no formato RGB

Com base nesta paleta, podemos definir os valores das seguintes cores básicas:

Cor	R	G	B	Valor
Preto	0	0	0	0
Azul	0	0	255	16711680
Verde	0	255	0	65280
Ciano	0	255	255	16776960
Vermelho	255	0	0	255
Rosa	255	0	255	16711935
Amarelo	255	255	0	65535
Branco	255	255	255	16777215

Para atribuir as cores aos objetos visuais devem ser observados os atributos utilizados para estes fins em cada objeto, como por exemplo:

MSDIALOG()

nClrPane	Cor de fundo do painel
nClrText	Cor da fonte das letras do painel

TSAY()

nClrPane	Cor de fundo do painel
nClrText	Cor da fonte das letras do painel

Função RGB()

A linguagem ADVPL possui a função RGB() a qual retorna o valor da cor a ser definido, de acordo com a parametrização de cada um dos elementos da paleta RGB.

RGB(nRed, nGreen, nBlue)

nRed	Valor de 0-255 para o elemento vermelho da paleta RGB
nGreen	Valor de 0-255 para o elemento verde da paleta RGB
nBlue	Valor de 0-255 para o elemento azul da paleta RGB
Retorno	Valor a ser definido para o atributo cor do componente

6. Aplicações com a interface visual do ADVPL

A linguagem ADVPL possui interfaces visuais pré-definidas que auxiliam no desenvolvimento de aplicações mais completas, combinando estas interfaces com os componentes visuais demonstrados anteriormente.

Didaticamente as interfaces visuais pré-definidas da linguagem ADVPL podem ser divididas em três grupos:

- Captura de informações simples ou Multi-Gets;
- Captura de múltiplas informações ou Multi-Lines;
- Barras de botões

6.1. Captura de informações simples (Multi-Gets)

Em ADVPL, as telas de captura de informações compostas por múltiplos campos digitáveis acompanhados de seus respectivos textos explicativos são comumente chamados de Enchoices.

Um Enchoice pode ser facilmente entendida como diversos conjuntos de objetos TSay e TGet alinhados de forma a visualizar ou capturar informações, normalmente vinculadas a arquivos de cadastros ou movimentações simples.

A linguagem ADVPL permite a implementação da Enchoice de duas formas similares:

- Função Enchoice: Sintaxe tradicionalmente utilizada em ADVPL, a qual não retorna um objeto para a aplicação chamadora;
- Classe MsMGet: Classe do objeto Enchoice, a qual permite a instanciação direta de um objeto, tornando-o disponível na aplicação chamadora.

A utilização de um ou outro objeto depende unicamente da escolha do desenvolvedor já que os parâmetros para a função Enchoice e para o método New() da classe MsMGet são os mesmos, lembrando que para manter a coerência com uma aplicação escrita em orientação a objetos deverá ser utilizada a classe MsMGet().

6.1.1. Enchoice()

- Sintaxe: Enchoice(cAlias, nReg, nOpc, aCRA, cLetra, cTexto, aAcho, aPos, aCpos, nModelo, nColMens, cMensagem, cTudoOk, oWnd, IF3, lMemoria, lColumn, caTela, lNoFolder, lProperty)
- Retorno: Nil
- Parâmetros:

cAlias	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
nReg	Parâmetro não utilizado

nOpc	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização)
aCRA	Parâmetro não utilizado
cLetra	Parâmetro não utilizado
cTexto	Parâmetro não utilizado
aAcho	Vetor com nome dos campos que serão exibidos. Os campos de usuário sempre serão exibidos se não existir no parâmetro um elemento com a expressão "NOUSER"
aPos	Vetor com coordenadas para criação da enchoice no formato {<top>, <left>, <bottom>, <right>}
aCpos	Vetor com nome dos campos que poderão ser editados
nModelo	Se for diferente de 1 desabilita execução de gatilhos estrangeiros
nColMens	Parâmetro não utilizado
cMensagem	Parâmetro não utilizado
cTudoOk	Expressão para validação da Enchoice
oWnd	Objeto (janela, painel, etc.) onde a enchoice será criada.
IF3	Indica se a enchoice esta sendo criada em uma consulta F3 para utilizar variáveis de memória
lMemoria	Indica se a enchoice utilizará variáveis de memória ou os campos da tabela na edição
lColumn	Indica se a apresentação dos campos será em forma de coluna
caTela	Nome da variável tipo "private" que a enchoice utilizará no lugar da propriedade aTela
lNoFolder	Indica se a enchoice não irá utilizar as Pastas de Cadastro (SXA)
lProperty	Indica se a enchoice não utilizará as variáveis aTela e aGets, somente suas propriedades com os mesmos nomes

Exemplo: Utilização da função Enchoice()

```
#include "protheus.ch"

/*/
+-----
| Função   | MBRWENCH   | Autor | ARNALDO RAYMUNDO JR. | Data |      |
+-----
| Descrição | Programa que demonstra a utilização da função Enchoice() |
+-----
| Uso       | Curso ADVPL
|
+-----
/*/
```

User Function MrbwEnch()

```
Private cCadastro           := " Cadastro de Clientes"
Private aRotina             :=      {"Pesquisar" , "axPesqui" , 0, 1},,
                               {"Visualizar" , "U_ModEnc" , 0, 2}}
```

DbSelectArea("SA1")

DbSetOrder(1)

MBrowse(6,1,22,75,"SA1")

Return

User Function ModEnc(cAlias,nReg,nOpc)

```
Local aCpoEnch             := {}
Local aAlter               := {}
Local cAliasE              := cAlias
Local aAlterEnch           := {}
Local aPos                 := {000,000,400,600}
Local nModelo              := 3
Local lF3                  := .F.
Local lMemoria             := .T.
Local lColumn              := .F.
Local caTela               := ""
Local lNoFolder            := .F.
Local lProperty            := .F.
Private oDlg
Private oGetD
Private oEnch
Private aTELA[0][0]
Private aGETS[0]
```

DbSelectArea("SX3")

DbSetOrder(1)

DbSeek(cAliasE)

```

While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
    If !(SX3->X3_CAMPO $ "A1_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And.;
        X3Uso(SX3->X3_USADO)

                                AADD(aCpoEnch,SX3->X3_CAMPO)

    EndIf
    DbSkip()
End

aAlterEnch := aClone(aCpoEnch)

DEFINE MSDIALOG oDlg TITLE cCadastro FROM 000,000 TO 400,600 PIXEL
    RegToMemory("SA1", If(nOpc==3,.T.,.F.))

    Enchoice(cAliasE, nReg, nOpc, /*aCRA*/, /*cLetra*/, /*cTexto*/, ;
        aCpoEnch, aPos, aAlterEnch, nModelo,

/*nColMens*/,;
        /*cMensagem*/,/*cTudoOk*/, oDlg, IF3, IMemoria, IColumn,;
        caTela, INoFolder, IProperty)
ACTIVATE MSDIALOG oDlg CENTERED
Return

```

6.1.2. MsMGet()

- Sintaxe: MsMGet():New(cAlias, nReg, nOpc, aCRA, cLetra, cTexto, aAcho, aPos, aCpos, nModelo, nColMens, cMensagem, cTudoOk, oWnd, IF3, IMemoria, IColumn, caTela, INoFolder, IProperty)
- Retorno: oMsMGet → objeto do tipo MsMGet()
- Parâmetros:

cAlias	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
nReg	Parâmetro não utilizado
nOpc	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização)
aCRA	Parâmetro não utilizado
cLetra	Parâmetro não utilizado
cTexto	Parâmetro não utilizado

aAcho	Vetor com nome dos campos que serão exibidos. Os campos de usuário sempre serão exibidos se não existir no parâmetro um elemento com a expressão "NOUSER"
aPos	Vetor com coordenadas para criação da enchoice no formato {<top>, <left>, <bottom>, <right>}
aCpos	Vetor com nome dos campos que poderão ser editados
nModelo	Se for diferente de 1 desabilita execução de gatilhos estrangeiros
nColMens	Parâmetro não utilizado
cMensagem	Parâmetro não utilizado
cTudoOk	Expressão para validação da Enchoice
oWnd	Objeto (janela, painel, etc.) onde a enchoice será criada.
IF3	Indica se a enchoice esta sendo criada em uma consulta F3 para utilizar variáveis de memória
IMemoria	Indica se a enchoice utilizará variáveis de memória ou os campos da tabela na edição
IColumn	Indica se a apresentação dos campos será em forma de coluna
caTela	Nome da variável tipo "private" que a enchoice utilizará no lugar da propriedade aTela
INoFolder	Indica se a enchoice não irá utilizar as Pastas de Cadastro (SXA)
IProperty	Indica se a enchoice não utilizará as variáveis aTela e aGets, somente suas propriedades com os mesmos nomes

Exemplo: Utilização do objeto MsMGet()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função   | MBRWMSGET | Autor | ARNALDO RAYMUNDO JR. | Data |      |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização do objeto MsMget() |
```

```
+-----
```

```
| Uso       | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
User Function MrbwMsGet()
```

```
Private cCadastro := " Cadastro de Clientes"
```

```
Private aRotina := {"Pesquisar" , "axPesqui" , 0, 1};;
```

```
 {"Visualizar" , "U_ModEnc" , 0, 2}}
```

```
DbSelectArea("SA1")
```

```
DbSetOrder(1)
```

```
MBrowse(6,1,22,75,"SA1")
```

```
Return
```

```
User Function ModEnc(cAlias,nReg,nOpc)
```

```
Local aCpoEnch := {}
```

```
Local aAlter := {}
```

```
Local cAliasE := cAlias
```

```
Local aAlterEnch := {}
```

```
Local aPos := {000,000,400,600}
```

```
Local nModelo := 3
```

```
Local lF3 := .F.
```

```
Local lMemoria := .T.
```

```
Local lColumn := .F.
```

```
Local caTela := ""
```

```
Local lNoFolder := .F.
```

```
Local lProperty := .F.
```

```
Private oDlg
```

```
Private oGetD
```

```
Private oEnch
```

```
Private aTELA[0][0]
```

```
Private aGETS[0]
```

```
DbSelectArea("SX3")
```

```
DbSetOrder(1)
```

```
DbSeek(cAliasE)
```

```
While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
```

```
    If !(SX3->X3_CAMPO $ "A1_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And.
```

```
    X3Uso(SX3->X3_USADO)
```

```

                                AADD(aCpoEnch,SX3->X3_CAMPO)

                                EndIf
                                DbSkip()

End

aAlterEnch := aClone(aCpoEnch)

oDlg      := MSDIALOG():New(000,000,400,600,cCadastro,,,,,,,,.T.)

RegToMemory(cAliasE, If(nOpc==3,.T.,.F.))







                                oEnch := MsMGet():New(cAliasE, nReg, nOpc, /*aCRA*/, /*cLetra*/,;
                                /*cTexto*/, aCpoEnch, aPos, aAlterEnch, nModelo, /*nColMens*/,;
                                /*cMensagem*/, /*cTudoOk*/,oDlg,IF3,IMemoria,IColumn, caTela,;
                                INoFolder, IProperty)
oDlg:ICentered := .T.
oDlg:Activate()

Return

```

6.2. Captura de múltiplas informações (Multi-Lines)

A linguagem ADVPL permite a utilização de basicamente dois tipos de objetos do tipo grid, ou como também são conhecidos: multi-line:

- Grids digitáveis: permitem a visualização e captura de informações, comumente utilizados em interfaces de cadastro e manutenção, tais como:
 -  MSGETDB()
 -  MSGETDADOS()
 -  MSNEWGETDADOS()
- Grids não digitáveis: permitem somente a visualização de informações, comumente utilizados como browses do ERP Protheus, tais como:
 -  TWBROWSE()
 -  MAWNDBROWSE()
 -  MBROWSE()

Neste tópico serão tratadas as grids digitáveis disponíveis na linguagem ADVPL para o desenvolvimento de interfaces de cadastros e manutenção de informações.

6.2.1. MsGetDB()

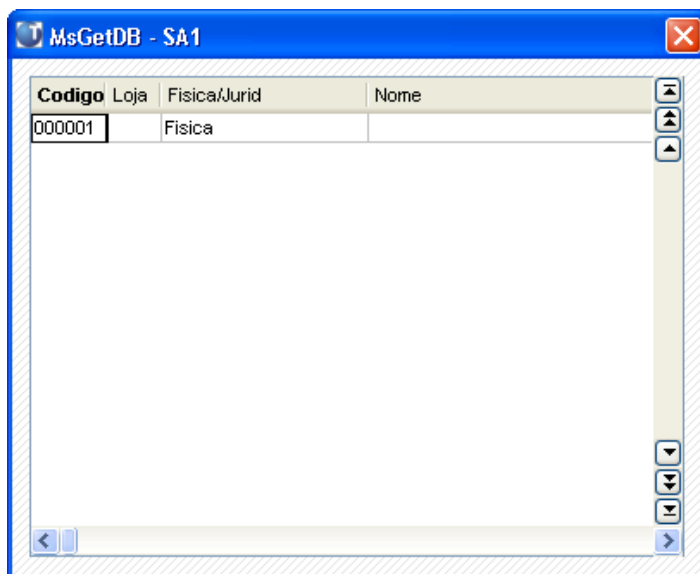
A classe de objetos visuais MsGetDB() permite a criação de um grid digitável com uma ou mais colunas, baseado em uma tabela temporária.

- Sintaxe: MsGetDB():New(nTop, nLeft, nBottom, nRight, nOpc, cLinhaOk, cTudoOk, cIniCpos, lDelete, aAlter, nFreeze, lEmpty, uPar1, cTRB, cFieldOk, lCondicional, lAppend, oWnd, lDisparos, uPar2, cDelOk, cSuperDel)
- Retorno: oMsGetDB() objeto do tipo MsGetDB()
- Parâmetros:

nTop	Distancia entre a MsGetDB e o extremidade superior do objeto que a contém.
nLeft	Distancia entre a MsGetDB e o extremidade esquerda do objeto que a contém.
nBottom	Distancia entre a MsGetDB e o extremidade inferior do objeto que a contém.
nRight	Distancia entre a MsGetDB e o extremidade direita do objeto que a contém.
nOpc	Posição do elemento do vetor aRotina que a MsGetDB usará como referência.
cLinhaOk	Função executada para validar o contexto da linha atual do aCols.
cTudoOk	Função executada para validar o contexto geral da MsGetDB (todo aCols).
cIniCpos	Nome dos campos do tipo caracter que utilizarão incremento automático. Este parâmetro deve ser no formato "+<nome do primeiro campo>+<nome do segundo campo>+...".
lDelete	Habilita a opção de excluir linhas do aCols. Valor padrão falso.
aAlter	Vetor com os campos que poderão ser alterados.
nFreeze	Indica qual coluna não ficara congelada na exibição.
lEmpty	Habilita validação da primeira coluna do aCols para esta não poder estar vazia. Valor padrão falso.
uPar1	Parâmetro reservado.
cFieldOk	Função executada na validação do campo.
cTRB	Alias da tabela temporária.
lCondicional	Reservado
lAppend	Indica se a MsGetDB ira criar uma linha em branco automaticamente quando for inclusão.
cDelOk	Função executada para validar a exclusão de uma linha do aCols.

IDisparos	Indica se será utilizado o Dicionário de Dados para consulta padrão, inicialização padrão e gatilhos.
uPar2	Parâmetro reservado.
cSuperDel	-Função executada quando pressionada as teclas <Ctrl>+<Delete>.
oWnd	Objeto no qual a MsGetDB será criada.

- Aparência:



- Variáveis private:

aRotina	<p>Vetor com as rotinas que serão executadas na MBrowse e que definira o tipo de operação que esta sendo executada (inclusão, alteração, exclusão, visualização, pesquisa, ...) no formato: {cTitulo, cRotina, nOpção, nAcesso}, aonde: nOpção segue o padrão do ERP Protheus para:</p> <ul style="list-style-type: none"> 1- Pesquisar 2- Visualizar 3- Incluir 4- Alterar 5- Excluir
aHeader	<p>Vetor com informações das colunas no formato:</p> <p>{cTitulo, cCampo, cPicture, nTamanho, nDecimais,; cValidação, cReservado, cTipo, xReservado1, xReservado2}</p> <p>A tabela temporária utilizada pela MsGetDB deverá ser criada com base no aHeader mais um último campo tipo lógico que determina se a linha foi excluída.</p>
lRefresh	Variável tipo lógico para uso reservado.

- Variáveis públicas:

nBrLin	Indica qual a linha posicionada do aCols.
---------------	---

- Funções de validação:

cLinhaOk	Função de validação na mudança das linhas da grid. Não pode ser definida como Static Function.
cTudoOk	Função de validação da confirmação da operação com o grid. Não pode ser definida como Static Function.

- Métodos adicionais:

ForceRefresh()	Atualiza a MsGetDB com a tabela e posiciona na primeira linha.
-----------------------	--

Exemplo: Utilização do objeto MsGetDB()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função | GETDBSA1 | Autor | MICROSIGA | Data | |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização do objeto MsGetDB() |
```

```
+-----
```

```
| Uso | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
User Function GetDbSA1()
```

```
Local nl
```

```
Local oDlg
```

```
Local oGetDB
```

```
Local nUsado := 0
```

```
Local aStruct := {}
```

```
Private lRefresh := .T.
```

```
Private aHeader := {}
```

```
Private aCols := {}
Private aRotina := {"Pesquisar", "AxPesqui", 0, 1},,
                  {"Visualizar", "AxVisual", 0, 2},,
                  {"Incluir", "AxInclui", 0, 3},,
                  {"Alterar", "AxAltera", 0, 4},,
                  {"Excluir", "AxDeleta", 0, 5}}

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek("SA1")

While !Eof() .and. SX3->X3_ARQUIVO == "SA1"
If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL
    nUsado++
    AADD(aHeader,{Trim(X3Titulo()),,
                SX3->X3_CAMPO,,
                SX3->X3_PICTURE,,

                SX3->X3_TAMANHO,,
                SX3->X3_DECIMAL,,
                SX3->X3_VALID,,
                ""},,
        SX3->X3_TIPO,,
        ""},,
        "" })
    AADD(aStruct,{SX3->X3_CAMPO,SX3->X3_TIPO,SX3->X3_TAMANHO,,
                  SX3->X3_DECIMAL})
EndIf

DbSkip()End

AADD(aStruct,{"FLAG","L",1,0})

cCriaTrab := CriaTrab(aStruct,.T.)
DbUseArea(.T.,__LocalDriver,cCriaTrab,,.T.,.F.)

oDlg      := MSDIALOG():New(000,000,300,400, "MsGetDB – SA1" ,,,,,,.T.)
```

```
oGetDB := MsGetDB():New(05,05,145,195,3,"U_LINHAOK", "U_TUDOOK", "+A1_COD", ;  
.T.,{"A1_NOME"},1,.F.,cCriaTrab,"U_FIELDOK",,.T.,oDlg, .T., "U_DELOK",;  
"U_SUPERDEL")
```

```
oDlg:ICentered := .T.  
oDlg:Activate()  
DbSelectArea(cCriaTrab)  
DbCloseArea()
```

```
Return
```

```
User Function LINHAOK()  
ApMsgStop("LINHAOK")  
Return .T.
```

```
User Function TUDOOK()  
ApMsgStop("LINHAOK")
```

```
Return .T.
```

```
User Function DELOK()  
ApMsgStop("DELOK")  
Return .T.
```

```
User Function SUPERDEL()  
ApMsgStop("SUPERDEL")  
Return .T.
```

```
User Function FIELDOK()  
ApMsgStop("FIELDOK")  
Return .T.
```

6.2.2. MsGetDados()

A classe de objetos visuais MsGetDados() permite a criação de um grid digitável com uma ou mais colunas, baseado em um array.

- Sintaxe: `MsGetDados():New(nTop, nLeft, nBottom, nRight, nOpc, cLinhaOk, cTudoOk, cIniCpos, lDelete, aAlter, uPar1, lEmpty, nMax, cFieldOk, cSuperDel, uPar2, cDelOk, oWnd)`
- Retorno: o`MsGetDados` é objeto do tipo `MsGetDados()`
- Parâmetros:

nTop	Distancia entre a <code>MsGetDados</code> e o extremidade superior do objeto que a contém.
nLeft	Distancia entre a <code>MsGetDados</code> e o extremidade esquerda do objeto que a contém.
nBottom	Distancia entre a <code>MsGetDados</code> e o extremidade inferior do objeto que a contém.
nRight	Distancia entre a <code>MsGetDados</code> e o extremidade direita do objeto que a contém.
nOpc	Posição do elemento do vetor <code>aRotina</code> que a <code>MsGetDados</code> usará como referencia.
cLinhaOk	Função executada para validar o contexto da linha atual do <code>aCols</code> .
cTudoOk	Função executada para validar o contexto geral da <code>MsGetDados</code> (todo <code>aCols</code>).
cIniCpos	Nome dos campos do tipo caracter que utilizarão incremento automático. Este parâmetro deve ser no formato “+<nome do primeiro campo>+<nome do segundo campo>+...”.
lDelete	Habilita excluir linhas do <code>aCols</code> . Valor padrão falso.
aAlter	Vetor com os campos que poderão ser alterados.
uPar1	Parâmetro reservado.
lEmpty	Habilita validação da primeira coluna do <code>aCols</code> para esta não poder estar vazia. Valor padrão falso.
nMax	Número máximo de linhas permitidas. Valor padrão 99.
cFieldOk	Função executada na validação do campo.
cSuperDel	Função executada quando pressionada as teclas <Ctrl>+<Delete>.
uPar2	Parâmetro reservado.
cDelOk	Função executada para validar a exclusão de uma linha do <code>aCols</code> .
oWnd	Objeto no qual a <code>MsGetDados</code> será criada.

- Aparência:



- Variáveis private:

aRotina	<p>Vetor com as rotinas que serão executadas na MBrowse e que definira o tipo de operação que esta sendo executada (inclusão, alteração, exclusão, visualização, pesquisa, ...) no formato:</p> <p>{cTitulo, cRotina, nOpção, nAcesso}, aonde:</p> <p>nOpção segue o padrão do ERP Protheus para:</p> <ol style="list-style-type: none"> 1- Pesquisar 2- Visualizar 3- Incluir 4- Alterar 5- Excluir
aHeader	<p>Vetor com informações das colunas no formato:</p> <p>{cTitulo, cCampo, cPicture, nTamanho, nDecimais,; cValidação, cReservado, cTipo, xReservado1, xReservado2}</p> <p>A tabela temporária utilizada pela MsGetDB deverá ser criada com base no aHeader mais um último campo tipo lógico que determina se a linha foi excluída.</p>
lRefresh	Variável tipo lógica para uso reservado.

- Variáveis públicas:

N	Indica qual a linha posicionada do aCols.
----------	---

- Funções de validação:

cLinhaOk	Função de validação na mudança das linhas da grid. Não pode ser definida como Static Function.
cTudoOk	Função de validação da confirmação da operação com o grid. Não pode ser definida como Static Function.

- Métodos adicionais:

ForceRefresh()	Atualiza a MsGetDados com a tabela e posiciona na primeira linha.
Hide()	Ocultar a MsGetDados.
Show()	Mostra a MsGetDados.

Exemplo: Utilização do objeto MsGetDados()

```
#include "protheus.ch"
```

```
/*
```

```
+-----
```

```
| Função   | GETDADOSA1 | Autor | MICROSIGA          | Data |      |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização do objeto MSGETADOS() |
```

```
+-----
```

```
| Uso       | Curso ADVPL
```

```
|
```

```
+-----
```

```
*/
```

```
User Function GetDadoSA1()
```

```
Local nl
```

```
Local oDlg
```

```
Local oGetDados
```

```
Local nUsado := 0
```

```
Private lRefresh := .T.
```

```
Private aHeader := {}
```

```
Private aCols := {}
```

```
Private aRotina := {"Pesquisar", "AxPesqui", 0, 1},;
```

```
                {"Visualizar", "AxVisual", 0, 2},;
```

```
                {"Incluir", "AxInclui", 0, 3},;
```

```

{"Alterar", "AxAlterar", 0, 4},;
{"Excluir", "AxDeleta", 0, 5}}

```

```

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek("SA1")

```

```

While !Eof() .and. SX3->X3_ARQUIVO == "SA1"
If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL

```

```

    nUsado++
    AADD(aHeader,{Trim(X3Titulo()),;
        SX3->X3_CAMPO,;
        SX3->X3_PICTURE,;
        SX3->X3_TAMANHO,;
        SX3->X3_DECIMAL,;
        SX3->X3_VALID,;
        "" ,;
        SX3->X3_TIPO,;
        "" ,;
        "" })

```

```

EndIf
DbSkip()
End

```

```

AADD(aCols,Array(nUsado+1))

```

```

For nl := 1 To nUsado
    aCols[1][nl] := CriaVar(aHeader[nl][2])
Next

```

```

aCols[1][nUsado+1] := .F.

```

```

oDlg      := MSDIALOG():New(000,000,300,400, "MsGetDados – SA1" ,,,,,,,.T.)

```



```
oGetDados := MsGetDados():New(05, 05, 145, 195, 4, "U_LINHAOK", "U_TUDOOK",;  
"+A1_COD", .T., {"A1_NOME"}, , .F., 200, "U_FIELDOK", "U_SUPERDEL",;  
"U_DELOK", oDlg)
```

```
oDlg:ICentered := .T.  
oDlg:Activate()
```

```
Return
```

```
User Function LINHAOK()  
ApMsgStop("LINHAOK")  
Return .T.
```

```
User Function TUDOOK()  
ApMsgStop("LINHAOK")  
Return .T.
```

```
User Function DELOK()  
ApMsgStop("DELOK")  
Return .T.
```

```
User Function SUPERDEL()  
ApMsgStop("SUPERDEL")  
Return .T.
```

```
User Function FIELDOK()  
ApMsgStop("FIELDOK")  
Return .T.
```

6.2.3. MsNewGetDados()

A classe de objetos visuais MsNewGetDados() permite a criação de um grid digitável com uma ou mais colunas, baseado em um array.

- Sintaxe: MsNewGetDados():New(nSuperior, nEsquerda ,nInferior, nDireita, nOpc, cLinOk, cTudoOk, cIniCpos, aAlterGDa, nFreeze, nMax, cFieldOk, cSuperDel, cDelOk, oDLG, aHeader, aCols)

- Retorno: oMsGetDados ? objeto do tipo MsNewGetDados()
- Parâmetros:

nSuperior	Distancia entre a MsNewGetDados e o extremidade superior do objeto que a contem
nEsquerda	Distancia entre a MsNewGetDados e o extremidade esquerda do objeto que a contem
nInferior	Distancia entre a MsNewGetDados e o extremidade inferior do objeto que a contem
nDireita	Distancia entre a MsNewGetDados e o extremidade direita do objeto que a contem
nOpc	Operação em execução: 2- Visualizar, 3- Incluir, 4- Alterar, 5- Excluir
cLinOk	Função executada para validar o contexto da linha atual do aCols
cTudoOk	Função executada para validar o contexto geral da MsNewGetDados (todo aCols)
clniCpos	Nome dos campos do tipo caracter que utilizarão incremento automático.
aAlterGDa	Campos alteráveis da GetDados
nFreeze	Campos estáticos na GetDados, partindo sempre da posição inicial da getdados aonde: 1- Primeiro campo congelado 2- Primeiro e segundo campos congelados...
nMax	Número máximo de linhas permitidas. Valor padrão 99
cFieldOk	Função executada na validação do campo
cSuperDel	Função executada quando pressionada as teclas <Ctrl>+<Delete>
cDelOk	Função executada para validar a exclusão de uma linha do aCols
oDLG	Objeto no qual a MsNewGetDados será criada
aHeader	Array a ser tratado internamente na MsNewGetDados como aHeader
aCols	Array a ser tratado internamente na MsNewGetDados como aCols

- Variáveis private:

aRotina	Vetor com as rotinas que serão executadas na MBrowse e que definira o tipo de operação que esta sendo executada (inclusão, alteração, exclusão, visualização, pesquisa, ...) no formato: {cTitulo, cRotina, nOpção, nAcesso}, aonde: nOpção segue o padrão do ERP Protheus para: 1- Pesquisar 2- Visualizar 3- Incluir 4- Alterar
----------------	---

aRotina	5- Excluir
aHeader	Vetor com informações das colunas no formato: {cTitulo, cCampo, cPicture, nTamanho, nDecimais,; cValidação, cReservado, cTipo, xReservado1, xReservado2} A tabela temporária utilizada pela MsGetDB deverá ser criada com base no aHeader mais um último campo tipo lógico que determina se a linha foi excluída.
lRefresh	Variável tipo lógica para uso reservado.

- Variáveis públicas:

N	Indica qual a linha posicionada do aCols.
----------	---

- Funções de validação:

cLinhaOk	Função de validação na mudança das linhas da grid. Não pode ser definida como Static Function.
cTudoOk	Função de validação da confirmação da operação com o grid. Não pode ser definida como Static Function.

- Métodos adicionais:

ForceRefresh()	Atualiza a MsNewGetDados com a tabela e posiciona na primeira linha.
Hide()	Oculto a MsNewGetDados.
Show()	Mostra a MsNewGetDados.

Exemplo: Utilização dos objetos MsNewGetDados() e MsMGet()

```
#include "protheus.ch"
/*
+-----+

| Função   | MBRWGETD | Autor | ARNALDO RAYMUNDO JR. | Data |      |
+-----+
| Descrição | Programa que demonstra a utilização dos objetos          |
|                                     | MsNewGetDados() e MsMGet() combinados
|                                     |
+-----+
| Uso       | Curso ADVPL
|
+-----+
```

```
/*/
```

```
User Function MrbwGetD()
```

```
Private cCadastro           := "Pedidos de Venda"
Private aRotina              := {"Pesquisar" , "axPesqui" , 0, 1},;
                             {"Visualizar" , "U_ModGtd", 0, 2},;
                             {"Incluir"      , "U_ModGtd", 0,
3}}
DbSelectArea("SC5")
DbSetOrder(1)
MBrowse(6,1,22,75,"SC5")
```

```
Return
```

```
User Function ModGtd(cAlias,nReg,nOpc)
```

```
Local nX                    := 0
Local nUsado                := 0
Local aButtons              := {}
Local aCpoEnch              := {}
Local cAliasE               := cAlias
Local aAlterEnch            := {}
Local aPos                  := {000,000,080,400}
Local nModelo               := 3
Local lF3                   := .F.

Local lMemoria              := .T.
Local lColumn               := .F.
Local caTela                := ""
Local lNoFolder             := .F.
Local lProperty             := .F.
Local aCpoGDa               := {}
Local cAliasGD              := "SC6"
Local nSuperior             := 081
Local nEsquerda             := 000
Local nInferior             := 250
Local nDireita              := 400
```

```

Local cLinOk           := "AlwaysTrue"
Local cTudoOk          := "AlwaysTrue"
Local cIniCpos         := "C6_ITEM"
Local nFreeze          := 000
Local nMax             := 999
Local cFieldOk         := "AlwaysTrue"
Local cSuperDel        := ""
Local cDelOk           := "AlwaysFalse"
Local aHeader          := {}
Local aCols            := {}
Local aAlterGDa        := {}

```

```

Private oDlg
Private oGetD
Private oEnch
Private aTELA[0][0]
Private aGETS[0]

```

```

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek(cAliasE)

```

```

While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
    If !(SX3->X3_CAMPO $ "C5_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And.;
X3Uso(SX3->X3_USADO)
                                AADD(aCpoEnch,SX3->X3_CAMPO)

```

```

        EndIf
        DbSkip()
End

```

```

aAlterEnch := aClone(aCpoEnch)

```

```

DbSelectArea("SX3")
DbSetOrder(1)

```

```

MsSeek(cAliasGD)

```

```

While !Eof() .And. SX3->X3_ARQUIVO == cAliasGD
    If
        !(AllTrim(SX3->X3_CAMPO) $ "C6_FILIAL") .And.;
        cNivel >= SX3->X3_NIVEL .And. X3Uso(SX3->X3_USADO)
        AADD(aCpoGDa,SX3->X3_CAMPO)
    EndIf
    DbSkip()
End

aAlterGDa := aClone(aCpoGDa)

nUsado:=0
dbSelectArea("SX3")
dbSeek("SC6")
aHeader:={}
While !Eof().And.(x3_arquivo=="SC6")
    If X3USO(x3_usado).And.cNivel>=x3_nivel
        nUsado:=nUsado+1
    AADD(aHeader,{ TRIM(x3_titulo), x3_campo, x3_picture,x3_tamanho,;
    x3_decimal,"AlwaysTrue()",x3_usado, x3_tipo, x3_arquivo, x3_context } )
    Endif
    dbSkip()
End

If nOpc==3 // Incluir
    aCols:={Array(nUsado+1)}
    aCols[1,nUsado+1]:=F.

    For nX:=1 to nUsado
        IF aHeader[nX,2] == "C6_ITEM"
            aCols[1,nX]:= "0001"
        ELSE
            aCols[1,nX]:=CriaVar(aHeader[nX,2])
        ENDIF
    Next
Else
    aCols:={}

```

```

dbSelectArea("SC6")
dbSetOrder(1)
dbSeek(xFilial()+M->C5_NUM)
While !eof().and.C6_NUM==M->C5_NUM
    AADD(aCols,Array(nUsado+1))
    For nX:=1 to nUsado

        aCols[Len(aCols),nX]:=FieldGet(FieldPos(aHeader[nX,2]))
        Next
        aCols[Len(aCols),nUsado+1]:=F.
        dbSkip()

    End
Endif

oDlg      := MSDIALOG():New(000,000,400,600, cCadastro,,,,,,,,T.)
RegToMemory("SC5", If(nOpc==3,.T.,.F.))

oEnch := MsMGet():New(cAliasE,nReg,nOpc,/*aCRA*/,/*cLetra*/,/*cTexto*/,;
                    aCpoEnch,aPos,aAlterEnch,          nModelo,
/*nColMens*/, /*cMensagem*/,;
/*cTudoOk*/, oDlg,IF3, lMemoria,lColumn,caTela,lNoFolder;;
lProperty)

oGetD:= MsNewGetDados():New(nSuperior, nEsquerda, nInferior, nDireita;;
nOpc,cLinOk,cTudoOk, cIniCpos, aAlterGDa, nFreeze, nMax,cFieldOk;;
cSuperDel,cDelOk, oDLG, aHeader, aCols)

oDlg:blnit := { | | EnchoiceBar(oDlg, { | | oDlg:End()},{ | | oDlg:End() },aButtons)}
oDlg:lCentered := .T.
oDlg:Activate()
Return

```

6.2.3.1. Definindo cores personalizadas para o objeto MsNewGetDados()

Conforme visto no tópico sobre definição das propriedades de cores para os componentes visuais, cada objeto possui características que devem ser respeitadas para correta utilização deste recurso.

- Atributos adicionais:

UseDefaultColors	Atributo que deverá ser definido como .F. para que as alterações nas cores sejam permitidas.
-------------------------	--

- Métodos adicionais:

SetBlkBackColor	Método que define a cor que será utilizada para cada linha do grid. Não é necessário utilizar o método Refresh() após a definição da cor por este método.
------------------------	---

Exemplo: Definindo cores personalizadas para o objeto MsNewGetDados()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função | MRBWGTCL | Autor | ARNALDO RAYMUNDO JR. | Data |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização dos  
objetos |  
| MsNewGetDados() e MsMGet() combinados e  
tratamento de cores |
```

```
+-----
```

```
| Uso | Curso ADVPL |
```

```
+-----
```

```
/*/
```

```
User Function MrbwGtCl()
```

```
Private cCadastro := "Pedidos de Venda"
```

```
Private aRotina := {"Pesquisar" , "axPesqui" , 0, 1},;  
{"Visualizar" , "U_ModGtd", 0, 2},;  
{"Incluir" , "U_ModGtd", 0,
```

```
3}}
```

```
DbSelectArea("SC5")
```

```
DbSetOrder(1)
```

```
MBrowse(6,1,22,75,"SC5")
```

```
Return
```


User Function ModGtd(cAlias,nReg,nOpc)

Local nX := 0

Local nUsado := 0

Local aButtons := {}

Local aCpoEnch := {}

Local cAliasE := cAlias

Local aAlterEnch := {}

Local aPos := {000,000,080,400}

Local nModelo := 3

Local IF3 := .F.

Local lMemoria := .T.

Local lColumn := .F.

Local caTela := ""

Local lNoFolder := .F.

Local lProperty := .F.

Local aCpoGDa := {}

Local cAliasGD := "SC6"

Local nSuperior := 081

Local nEsquerda := 000

Local nInferior := 250

Local nDireita := 400

Local cLinOk := "AlwaysTrue"

Local cTudoOk := "AlwaysTrue"

Local cIniCpos := "C6_ITEM"

Local nFreeze := 000

Local nMax := 999

Local cFieldOk := "AlwaysTrue"

Local cSuperDel := ""

Local cDelOk := "AlwaysFalse"

Local aHeader := {}

Local aCols := {}

Local aAlterGDa := {}

Private oDlg

Private oGetD

Private oEnch

Private aTELA[0][0]

Private aGETS[0]

DbSelectArea("SX3")

DbSetOrder(1)

DbSeek(cAliasE)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasE

 If !(SX3->X3_CAMPO \$ "C5_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And.;

 X3Uso(SX3->X3_USADO)

 AADD(aCpoEnch,SX3->X3_CAMPO)

 EndIf

 DbSkip()

End

aAlterEnch := aClone(aCpoEnch)

DbSelectArea("SX3")

DbSetOrder(1)

MsSeek(cAliasGD)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasGD

 If !(AllTrim(SX3->X3_CAMPO) \$ "C6_FILIAL") .And.

 cNivel >= SX3->X3_NIVEL .And. X3Uso(SX3->X3_USADO)

 AADD(aCpoGDa,SX3->X3_CAMPO)

 EndIf

 DbSkip()

End

aAlterGDa := aClone(aCpoGDa)

nUsado:=0

dbSelectArea("SX3")

dbSeek("SC6")

```

aHeader:={}
While !Eof().And.(x3_arquivo=="SC6")
    If X3USO(x3_usado).And.cNivel>=x3_nivel
        nUsado:=nUsado+1
    AADD(aHeader,{ TRIM(x3_titulo), x3_campo, x3_picture,;
        x3_tamanho, x3_decimal,"AlwaysTrue()";;
        x3_usado, x3_tipo, x3_arquivo, x3_context } )
    Endif
dbSkip()
End

If nOpc==3 // Incluir
    aCols:={Array(nUsado+1)}
    aCols[1,nUsado+1]:=F.
    For nX:=1 to nUsado

        IF aHeader[nX,2] == "C6_ITEM"
            aCols[1,nX]:= "0001"
        ELSE
            aCols[1,nX]:=CriaVar(aHeader[nX,2])
        ENDIF

    Next

Else
    aCols:={}
    dbSelectArea("SC6")
    dbSetOrder(1)
    dbSeek(xFilial()+M->C5_NUM)
    While !eof().and.C6_NUM==M->C5_NUM
        AADD(aCols,Array(nUsado+1))
        For nX:=1 to nUsado

            aCols[Len(aCols),nX]:=FieldGet(FieldPos(aHeader[nX,2]))

        Next
        aCols[Len(aCols),nUsado+1]:=F.
        dbSkip()
    End
End

```

Endif

```
oDlg := MSDIALOG():New(000,000,400,600, cCadastro,,,,,,,,,T.)
```

```
RegToMemory("SC5", If(nOpc==3,.T.,.F.))
```

```
oEnch := MsMGet():New(cAliasE,nReg,nOpc,/*aCRA*/,/*cLetra*/, /*cTexto*/,,
                    aCpoEnch,aPos,    aAlterEnch,    nModelo,
/*nColMens*/, /*cMensagem*/,,
                    cTudoOk,oDlg,IF3,
IMemoria,IColumn,caTela,IInoFolder,IProperty)
```

```
oGetD:= MsNewGetDados():New(nSuperior,nEsquerda,nInferior,nDireita, nOpc,,
cLinOk,cTudoOk,clniCpos,aAlterGDa,nFreeze,nMax,cFieldOk, cSuperDel,,
cDelOk, oDLG, aHeader, aCols)
```

```
// Tratamento para definição de cores específicas,
// logo após a declaração da MsNewGetDados
```

```
oGetD:oBrowse:UseDefaultColors := .F.
oGetD:oBrowse:SetBlkBackColor({| | GETDCLR(oGetD:aCols,oGetD:nAt,aHeader)})
```

```
oDlg:blnit := {| | EnchoiceBar(oDlg, {| | oDlg:End()), {| | oDlg:End()),aButtons}}
oDlg:Icentered := .T.
oDlg:Activate()
```

Return

```
// Função para tratamento das regras de cores para a grid da MsNewGetDados
```

```
Static Function GETDCLR(aLinha,nLinha,aHeader)
```

```
Local nCor2                := 16776960 // Ciano - RGB(0,255,255)
Local nCor3                := 16777215 // Branco - RGB(255,255,255)
Local nPosProd              := aScan(aHeader,{|x| Alltrim(x[2]) ==
"C6_PRODUTO"})
Local nUsado                := Len(aHeader)+1
```

```
Local nRet                                := nCor3

If !Empty(aLinha[nLinha][nPosProd]) .AND. aLinha[nLinha][nUsado]
    nRet := nCor2
Elseif !Empty(aLinha[nLinha][nPosProd]) .AND. !aLinha[nLinha][nUsado]
    nRet := nCor3
Endif

Return nRet
```

6.3. Barras de botões

A linguagem ADVPL permite a implementação de barras de botões utilizando funções pré-definidas desenvolvidas com o objetivo de facilitar sua utilização, ou através da utilização direta dos componentes visuais disponíveis. Dentre os recursos da linguagem que podem ser utilizados com esta finalidade serão abordados:

- Função EnchoiceBar: Sintaxe tradicionalmente utilizada em ADVPL, a qual não retorna um objeto para a aplicação chamadora;
- Classe TBar: Classe do objeto TBar(), a qual permite a instanciação direta de um objeto do tipo barra de botões superior, tornando-o disponível na aplicação chamadora.
- Classe ButtonBar: Classe do objeto ButtonBar(), a qual permite a instanciação direta de um objeto barra de botões genérico, o qual pode ser utilizado em qualquer posição da tela, tornando-o disponível na aplicação chamadora.

6.3.1. EnchoiceBar()

Função que cria uma barra de botões no formato padrão utilizado pelas interfaces de cadastro da aplicação Protheus.

Esta barra possui os botões padrões para confirmar ou cancelar a interface e ainda permite a adição de botões adicionais com a utilização do parâmetro aButtons.

- Sintaxe:
EnchoiceBar(oDlg, bOk, bCancel, lMsgDel, aButtons, nRecno, cAlias)
- Parâmetros:

oDlg	Dialog onde irá criar a barra de botões
bOk	Bloco de código a ser executado no botão Ok
bCancel	Bloco de código a ser executado no botão Cancelar

lMsgDel	Exibe dialog para confirmar a exclusão
aButtons	Array contendo botões adicionais. aArray[n][1] -> Imagem do botão aArray[n][2] -> bloco de código contendo a ação do botão aArray[n][3] -> título do botão
nRecno	Registro a ser posicionado após a execução do botão Ok.
cAlias	Alias do registro a ser posicionado após a execução do botão Ok. Se o parâmetro nRecno for informado, o cAlias passa ser obrigatório.

Exemplo: Utilização da função EnchoiceBar()

```
#include "protheus.ch"
```

```
/*/
```

```
+-----
```

```
| Função   | DENCHBAR   | Autor | ARNALDO RAYMUNDO JR. | Data |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização da função
```

```
| EnchoiceBar()
```

```
|
```

```
+-----
```

```
| Uso      | Curso ADVPL
```

```
|
```

```
+-----
```

```
/*/
```

```
User Function DEnchBar()
```

```
Local oDlg, oBtn
```

```
Local aButtons := {}
```

```
DEFINE MSDIALOG oDlg TITLE "Teste EnchoiceBar" FROM 000,000 TO 400,600 PIXEL OF;
```

```
oMainWnd
```

```
AADD( aButtons, {"HISTORIC", {| | TestHist()}, "Histórico...";
```

```
      "Histórico",{| | .T.} )
```

```
@ -15,-15 BUTTON oBtn PROMPT "..." SIZE 1,1 PIXEL OF oDlg
```

```
ACTIVATE MSDIALOG oDlg ;
```

```
ON INIT (EnchoiceBar(oDlg,{| | IOk:=.T.,oDlg:End()},{| | oDlg:End()},@aButtons))
```

```
Return
```

6.3.2.TBar()

Classe de objetos visuais que permite a implementação de um componente do tipo barra de botões para a parte superior de uma janela previamente definida.

- Sintaxe: New(oWnd, nBtnWidth, nBtnHeight, l3D, cMode, oCursor, cResource, lNoAutoAdjust)
- Retorno: oTBar → objeto do tipo TBar()
- Parâmetros:

oWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
nBtnWidth	Numérico, opcional. Largura do botão contido na barra
nBtnHeight	Numérico, opcional. Altura do botão contido na barra
l3D	Lógico, opcional. Define tipo da barra
cMode	Não utilizado.
oCursor	Objeto, opcional. Define Cursor ao posicionar o mouse sobre a barra.
cResource	Caracter, opcional. Imagem do recurso a ser inserido como fundo da barra.
lNoAutoAdjust	Lógico.

Exemplo: Utilização da função EnchoiceBar()

```
#include 'protheus.ch'
```

```
/*/
```

```
+-----+
| Função   | TSTBAR   | Autor | MICROSIGA | Data |   |
+-----+
| Descrição | Programa que demonstra a utilização do objeto TBar() |
+-----+
| Uso       | Curso ADVPL
|           |
+-----+
```

```
/*/
```

```
User Function TstTBar()
```

```
Local oDlg
```

```
oDlg      := MSDIALOG():New(000,000,305,505, 'Exemplo - TBar' ,,,,,,,,,.T.)
```

```
oTBar := TBar():New( oDlg,25,32,.T.,,,,F. )
```

```
oTBtnBmp2_1                               := TBtnBmp2():New( 00, 00, 35, 25, 'copyuser' ,,,,;
{| |Alert('TBtnBmp2_1')}, oTBar,'msGetEx' ,,,F.,.F. )
```

```
oTBtnBmp2_2                               := TBtnBmp2():New( 00, 00, 35, 25, 'critica' ,,,,;
{| |},oTBar,'Critica' ,,,F.,.F. )
```

```
oTBtnBmp2_3                               := TBtnBmp2():New( 00, 00, 35, 25, 'bmpcpo' ,,,,;
{| |},oTBar,'PCO' ,,,F.,.F. )
```

```
oTBtnBmp2_4                               := TBtnBmp2():New( 00, 00, 35, 25, 'preco' ,,,,;
{| |},oTBar,'Preço' ,,,F.,.F. )
```

```
oDlg:ICentered := .T.
```

```
oDlg:Activate()
```

```
Return
```

6.3.3. ButtonBar

A sintaxe ButtonBar é a forma clássica utilizada na linguagem ADVPL para implementar um objeto da classe TBar(), o qual possui as características mencionadas no tópico anterior.

- Sintaxe:

```
DEFINE BUTTONBAR oBar SIZE nWidth, nHeight 3D MODE OF oDlg  CURSOR
```

- Retorno: ().

- Parâmetros:

oBar	Objeto do tipo TBar() que será criado com a utilização da sintaxe ButtonBar().
nWidth	Numérico, opcional. Largura do botão contido na barra.
nHeight	Numérico, opcional. Altura do botão contido na barra.
3D	Se definido habilita a visualização em 3D da barra de botões.
oDlg	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
MODE	Define a forma de orientação do objeto ButtonBar utilizando os seguintes termos pré-definidos: TOP, BOTTOM, FLOAT
CURSOR	Objeto, opcional. Define Cursor ao posicionar o mouse sobre a barra.

A sintaxe ButtonBar requer a adição dos botões como recursos adicionais da barra previamente definida utilizando a sintaxe abaixo:

- Botões: BUTTON RESOURCE
- Sintaxe adicional:

DEFINE BUTTON RESOURCE cBitMap OF oBar ACTION cAcao TOOLTIP cTexto

- Parâmetros:

cBitMap	Nome da imagem disponível na aplicação.
oBar	Objeto do tipo TBar() no qual o botão será adicionado.
cAcao	Função ou lista de expressões que determina a ação que será realizada pelo botão.
cTexto	Texto no estilo "tooltip text" que será exibido quando o cursor do mouse for posicionado sobre o botão na barra de ferramentas.

Exemplo: Utilização da sintaxe ButtonBar

```
#include 'protheus.ch'
```

```
/*/
```

```
+-----
```

```
| Função   | TstBBar   | Autor | MICROSIGA           | Data |   |
```

```
+-----
```

```
| Descrição | Programa que demonstra a utilização do objeto TBar() |
```

```
+-----
```

```
| Uso       | Curso ADVPL                                     |
```

```
+-----
```

```
/*/
```

```
User Function TstBBar()
```

```
Local oDlg
```

```
Local oBtn1
```

```
Local oBtn2
```

```
oDlg      := MSDIALOG():New(000,000,305,505, 'Exemplo - BUTTONBAR',,,,,,,,,T.)
```

```
DEFINE BUTTONBAR oBar SIZE 25,25 3D TOP OF oDlg
```

```

DEFINE BUTTON RESOURCE "S4WB005N"    OF oBar ACTION NaoDisp() TOOLTIP "Recortar"
DEFINE BUTTON RESOURCE "S4WB006N"    OF oBar ACTION NaoDisp() TOOLTIP "Copiar"
DEFINE BUTTON RESOURCE "S4WB007N"    OF oBar ACTION NaoDisp() TOOLTIP "Colar"
DEFINE BUTTON oBtn1 RESOURCE "S4WB008N" OF oBar GROUP;
ACTION Calculadora() TOOLTIP "Calculadora"
oBtn1:cTitle:="Calc"
DEFINE BUTTON RESOURCE "S4WB009N"    OF oBar ACTION Agenda() TOOLTIP "Agenda"
DEFINE BUTTON RESOURCE "S4WB010N"    OF oBar ACTION OurSpool() TOOLTIP "Spool"
DEFINE BUTTON RESOURCE "S4WB016N"    OF oBar GROUP;
ACTION HelProg() TOOLTIP "Ajuda"
DEFINE BUTTON oBtn2 RESOURCE "PARAMETROS" OF oBar GROUP;
ACTION Sx1C020() TOOLTIP "Parâmetros"
oBtn2:cTitle:="Param."
DEFINE    BUTTON oBtOk RESOURCE "FINAL" OF oBar GROUP;
ACTION oDlg:End()TOOLTIP "Sair"
oBar:bRClicked := { | | AlwaysTrue() }
oDlg:lcCentered := .T.
oDlg:Activate()
Return

```

6.3.4. Imagens pré-definidas para as barras de botões






Conforme mencionado nos tópicos anteriores, os botões visuais do tipo barra de botões permitem a definição de itens com ações e imagens vinculadas.

Dentre os objetos e funções mencionados, foi citada a `EnchoiceBar()`, a qual permite a adição de botões adicionais através do parâmetro `aButton`, sendo que os itens deste array devem possuir o seguinte formato:

- Sintaxe: `AADD(aButtons,{cBitMap, bAcao, cTexto})`
- Estrutura:

cBitMap	Nome da imagem pré-definida existente na aplicação ERP que será vinculada ao botão.
bAcao	Bloco de código que define a ação que será executada com a utilização do botão.
cTexto	Texto no estilo "tooltip text" que será exibido quando o cursor do mouse for posicionado sobre o botão na barra de ferramentas.

- Alguns BitMaps disponíveis:

	DESTINOS		DISCAGEM
	EDIT		EDITABLE
	EXCLUIR		FORM
	GRAF2D		GRAF3D
	LINE		NOTE
	OBJETIVO		OK
	PENDENTE		PRECO
	PRODUTO		S4SB014N
	S4WB001N		S4WB005N
	S4WB006N		S4WB007N
	S4WB008N		S4WB009N
	S4WB010N		S4WB011N
	S4WB013N		S4WB014A
	S4WB016N		SIMULACA
	VENDEDOR		USER

- Exemplo:

```
AADD(aButtons,{"USER",{"| | AllwaysTrue()"},"Usuário"})
```

APÊNDICES

BOAS PRÁTICAS DE PROGRAMAÇÃO

7. Arredondamento

Algumas operações numéricas podem causar diferenças de arredondamento. Isso ocorre devido a diferenças no armazenamento de variáveis numéricas nos diversos processadores, diferença esta, inclusive, presente no ADVPL, mesmo antes do surgimento do Protheus.

Para evitar esses problemas de arredondamento, deve ser utilizada a função Round(), principalmente antes de realizar uma comparação e antes de se utilizar a função Int().

Desse modo, assegura-se que o resultado será correto independentemente do processador ou plataforma.

Exemplo 01:

```
If (Valor/30) == 50      // pode ser falso ou inválido
If Round(Valor/30, 0) == 50 // correto
```

Exemplo 02:

```
M->EE8_QTDEM1 := Int(M->EE8_SLDINI/M->EE8_QE) // pode ser falso ou inválido
M->EE8_QTDEM1 := Int(Round(M->EE8_SLDINI/M->EE8_QE,10)) // correto
```

8. Utilização de Identação

É obrigatória a utilização da indentação, pois torna o código muito mais legível. Veja os exemplos abaixo:

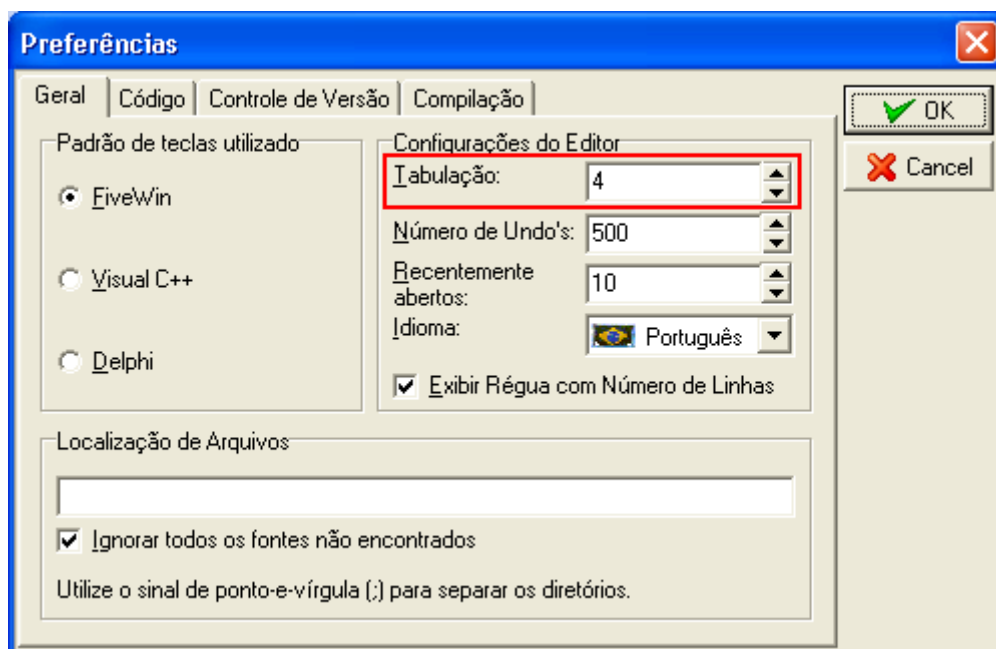
```
While !SB1->(Eof())
If mv_par01 == SB1->B1_COD
dbSkip()
Loop
Endif
Do Case
Case SB1->B1_LOCAL == "01" .OR. SB1->B1_LOCAL == "02"
TrataLocal(SB1->B1_COD, SB1->B1_LOCAL)
Case SB1->B1_LOCAL == "03"
TrataDefeito(SB1->B1_COD)
OtherWise
TrataCompra(SB1->B1_COD, SB1->B1_LOCAL)
EndCase
dbSkip()
EndDo
```

A utilização da indentação seguindo as estruturas de controle de fluxo (while, IF, caso etc.) torna a compreensão do código muito mais fácil:

```
While !SB1->(Eof())
    If mv_par01 == SB1->B1_COD
        dbSkip()
    Loop
Endif
Do Case
    Case SB1->B1_LOCAL == "01" .OR. SB1->B1_LOCAL == "02"
        TrataLocal(SB1->B1_COD, SB1->B1_LOCAL)
    Case SB1->B1_LOCAL == "03"
        TrataDefeito(SB1->B1_COD)
    OtherWise
        TrataCompra(SB1->B1_COD, SB1->B1_LOCAL)
EndCase
dbSkip()

EndDo
```

Para indentar o código utilize a tecla <TAB> e na ferramenta DEV-Studio, a qual pode ser configurada através da opção “Preferências”:



9. Capitulação de Palavras-Chave

Uma convenção amplamente utilizada é a de capitular as palavras chaves, funções, variáveis e campos utilizando uma combinação de caracteres em maiúsculo e minúsculo, visando facilitar a leitura do código fonte. O código a seguir:

```
Local ncnt while ( ncnt++ < 10 ) ntotal += ncnt * 2 enddo
```

Ficaria melhor com as palavras chaves e variáveis capituladas:

```
Local nCnt While ( nCnt++ < 10 ) nTotal += nCnt * 2 EndDo
```

Importante

Para funções de manipulação de dados que comecem por “db”, a capitulação só será efetuada após o “db”:

- dbSeek()
- dbSelectArea()

9.1. Palavras em maiúsculo

A regra é utilizar caracteres em maiúsculo para:

- Constantes:

```
#define NUMLINES 60 #define NUMPAGES 1000
```

- Variáveis de memória:

```
M-> CT2_CRCONV M->CT2_MCONVER := CriaVar("CT2_CONVER")
```

- Campos:

```
SC6->C6_NUMPED
```

- Querys:

```
SELECT * FROM...
```

10. Utilização da Notação Húngara

A notação húngara consiste em adicionar os prefixos aos nomes de variáveis, de modo a facilmente se identificar seu tipo. Isto facilita na criação de códigos-fonte extensos, pois usando a Notação Húngara, você não precisa ficar o tempo todo voltando à definição de uma variável para se lembrar qual é o tipo de dados que deve ser colocado nela. Variáveis devem ter um prefixo de Notação Húngara em minúsculas, seguido de um nome que identifique a função da variável, sendo que a inicial de cada palavra deve ser maiúscula.

É obrigatória a utilização desta notação para nomear variáveis.

Notação	Tipo de dado	Exemplo
a	Array	aValores
b	Bloco de código	bSeek
c	Caracter	cNome
d	Data	dDataBase
l	Lógico	lContinua
n	Numérico	nValor
o	Objeto	oMainWindow
x	Indefinido	xConteudo

11. Técnicas de programação eficiente

Para o desenvolvimento de sistemas e a programação de rotinas, sempre é esperado que qualquer código escrito seja:

- Funcionalmente correto
- Eficiente
- Legível
- Reutilizável
- Extensível
- Portável

Após anos de experiência na utilização de linguagens padrão xBase e do desenvolvimento da linguagem ADVPL, algumas técnicas para uma programação otimizada e eficiente foram reconhecidas. A utilização das técnicas a seguir, visa buscar o máximo aproveitamento dos recursos da linguagem com o objetivo de criar programas com estas características.

Criação de funções segundo a necessidade

Observe o código de exemplo:

```
User Function GetAnswer(IDefault)
Local IOk
IOk := GetOk(IDefault)
If IOk
    Return .T.
Else
    Return .F.
Endif
Return nil
```

Utilizando-se apenas o critério "a função funciona corretamente?", a função GetAnswer é perfeita. Recebe um parâmetro lógico com a resposta padrão e retorna um valor lógico dependente da opção escolhida pelo usuário em uma função de diálogo "sim/não" designada para isso. Pode entretanto ser melhorada, particularmente se eficiência for considerada como um critério para um código melhor. Eficiência tipicamente envolve a utilização de poucos recursos de máquina, poucas chamadas de funções ou tornar mais rápido um processo.

Segundo esse raciocínio, poderia se produzir o seguinte código:

```
User Function GetAnswer(IDefault)
Return If( GetOk(IDefault), .T., .F.)
```

O código acima ainda pode ser aprimorado conforme abaixo:

```
User Function GetAnswer(IDefault)
Return GetOk(IDefault)
```

Com a otimização do código da função `GetAnswer()`, pode facilmente verificar que a mesma não realiza nada adicional à chamada de `GetOk()`, podendo ser substituída por uma chamada direta desta, continuando a funcionar corretamente.

Codificação auto-documentável

Nenhum comentário substitui um código claramente escrito, e este não é um acidente. Considere o exemplo:

```
cVar := "          " // 11 espaços
```

O tamanho da variável `cVar` não é evidente por si só e não é facilmente verificado. Estes mesmos 10 espaços estariam mais óbvios e ainda assim garantidos se a instrução fosse escrita como:

```
cVar := Space(11)
```

O mesmo princípio pode ser aplicado para qualquer string longa de caracteres repetidos. A função `Replicate` pode ser utilizada como a seguir:

```
cVar := Replicate( "*", 80 )
```

Este tipo de programação deixa o código fácil de digitar, fácil de ler e mais flexível.

Utilização de soluções simples

Simplicidade na criação de instruções torna a programação e até mesmo a execução mais rápida. Considere a linha de código:

```
If nVar > 0 .Or. nVar < 0
```

Se o valor da variável `nVar` for igual a zero (0) no momento da execução desta linha de código, ambas as comparações separadas pelo operador lógico `.Or.` serão efetuadas: Após ser avaliada, a primeira comparação irá falhar. A segunda comparação será então avaliada e falhará também. Como resultado, o código existente dentro da estrutura de fluxo `If` não será executado. Tal código somente será executado quando o valor desta variável for maior OU menor do que zero. Ou seja, sempre que for DIFERENTE de zero, o que torna a linha a seguir mais eficiente:

```
If nVar != 0
```

Este tipo de alteração torna o código mais legível e o processamento mais rápido, evitando a avaliação de instruções desnecessariamente.

Existem outras situações onde a simplificação pode ser utilizada. A expressão de avaliação a seguir:

```
If cVar == "A" .Or. cVar == "B" .Or cVar == "C" .Or. cVar == "D"
```

Pode ser substituído pelo operador de contenção:

```
If cVar $ "ABCD"
```

Opção por flexibilidade

A melhor solução é aquela que envolve o problema imediato e previne problemas no futuro. Considere o exemplo:

```
@nRow,nCol PSAY cVar Picture "!!!!!!!!!!!!!!!!!!!!!!"
```

Exceto contando-se os caracteres, não existe maneira de saber se o número de caracteres de exclamação é o esperado. Enquanto isto é um problema, existem algo mais grave. A expressão de picture é estática. Se no futuro for necessário ajustar o tamanho da variável cVar, será necessário localizar todos os lugares no código onde esta máscara de picture está sendo utilizada para ajuste manual.

Existe uma opção de solução de auto-ajuste disponível que é fácil de digitar e tem a garantia de executar a tarefa igualmente (tornar todos os caracteres maiúsculos):

```
@nRow,nCol PSAY cVar Picture "@!"
```

Opção da praticidade ao drama

Se a solução parece complexa, provavelmente é porque o caminho escolhido está levando a isso. Deve-se sempre se perguntar porque alguém desenvolveria uma linguagem que requisite tantos comandos complicados para fazer algo simples. Na grande maioria dos casos, existe uma solução mais simples. O exemplo abaixo deixa isso bem claro:

```
@ 10,25 Say Substr(cCep,1,5) + "-" + Substr(cCep,6,3) Picture "!!!!!!!!!!"
```

Este código pode ser escrito de uma forma muito mais simples, conforme demonstrado abaixo:

```
@ 10,25 Say cCep Picture "@R 99999-999"
```

Utilização de operadores de incremento/decremento

Utilizados devidamente, os operadores de incremento e decremento tornam o código mais fácil de ler e possivelmente um pouco mais rápidos. Ao contrário de escrever adições simples como:

```
nVar := nVar + 1
```

```
nVar := nVar -1
```

Pode-se escrevê-las assim:

```
++nVar  
--nVar
```

Deve-se apenas tomar cuidado com a precedência destes operadores, pois o "++" ou o "--" podem aparecer antes ou depois de uma variável, e em alguns casos quando a variável for utilizada dentro de uma expressão, a prefixação ou sufixação destes operadores afetará o resultado. Para maiores detalhes, consulte a documentação de operadores da linguagem ADVPL.

Evitar passos desnecessários

Existe uma diferença entre um bom hábito e perda de tempo. Algumas vezes estes conceitos podem estar muito próximos, mas um modo de diferenciá-los é balancear os benefícios de realizar alguma ação contra o problema que resultaria se não fosse executada. Observe o exemplo:

```
Local nCnt := 0  
For nCnt := 1 To 10  
<código>  
Next nCnt
```

Inicializar a variável no momento da declaração não é um problema. Se o 0 fosse necessário no exemplo, teria sido útil a inicialização na declaração. Mas neste caso a estrutura de repetição For...Next atribui o seu valor imediatamente com 1, portanto não houve ganho em atribuir a variável com 0 no começo.

Neste exemplo não há nenhum ponto negativo e nada errado ocorrerá se a variável não for inicializada, portanto é aconselhável evitar este tipo de inicialização, pois não torna o código mais seguro e também não expressa a intenção do código mais claramente.

Porém note este exemplo, onde a variável não é inicializada:

```
Local nCnt  
While ( nCnt++ < 10 )  
<código>  
EndDo
```

Em ADVPL, variáveis não inicializadas sempre tem seu valor contendo nulo (nil) a princípio, o que fará com que uma exceção em tempo de execução aconteça quando a instrução de repetição while for executada.

Diferentemente do primeiro exemplo, onde a inicialização da variável não fazia diferença alguma, neste segundo exemplo a inicialização é absolutamente necessária. Deve-se procurar inicializar variáveis numéricas com zero (0) e variáveis caracter com string nula (""), apenas quando realmente necessário.

Utilização de alternativas

Quando se está trabalhando em uma simples rotina, deve-se tomar algum tempo para explorar duas ou três diferentes abordagens. Quando se está trabalhando em algo mais complexo, deve-se planejar prototipar algumas a mais. Considere o seguinte código:

```
If cHair = "A"
  Replace hair With "Loira"
Else
  If cHair = "B"
    Replace hair With "Morena"
  Else
    If cHair = "C"
      Replace hair With "Ruiva"
    Else
      If cHair = "D"
        Replace hair With "Grisalho"
      Else
        Replace hair With "Preto"
      Endif
    Endif
  Endif
Endif
```

Um código de uma única letra, (A até E), foi informado para indicar a cor de cabelo. Este código foi então convertido e armazenado como uma string. Pode-se notar que a cor "Preto" será atribuída se nenhuma outra opção for verdadeira.

Uma alternativa que reduz o nível de indentação torna o código mais fácil de ler enquanto reduz o número de comandos replace:

```
Do Case
  Case cHair == "A"
    cColor := "Loira"
  Case cHair == "B"
    cColor := "Morena"
  Case cHair == "C"
    cColor := "Ruiva"
  Case cHair == "D"
    cColor := "Grisalho"
```

```
OtherWise
    cColor := "Preto"
EndCase

Replace hair With cColor
```

Utilização de arquivos de cabeçalho quando necessário

Se um arquivo de código criado se referencia a comandos para interpretação e tratamento de arquivos XML, este deve se incluir o arquivo de cabeçalho próprio para tais comandos (XMLXFUN.CH no exemplo). Porém não deve-se incluir arquivos de cabeçalho apenas por segurança. Se não se está referenciando nenhuma das constantes ou utilizando nenhum dos comandos contidos em um destes arquivos, a inclusão apenas tornará a compilação mais demorada.

Constantes em maiúsculo

Isto é uma convenção que faz sentido. Em ADVPL, como em C por exemplo, a regra é utilizar todos os caracteres de uma constante em maiúsculo, a fim de que possam ser claramente reconhecidos como constantes no código, e que não seja necessários lembrar onde foram declarados.

Utilização de indentação

Este é um hábito que todo programador deve desenvolver. Não consome muito esforço para manter o código alinhado durante o trabalho, porém quando necessário pode-se utilizar a ferramenta TOTVS DevStudio para a re-indentação de código. Para maiores detalhes, consulte a documentação sobre a indentação de códigos fontes disponível nos demais tópicos deste material.

Utilização de espaços em branco

Espaços em branco extras tornam o código mais fácil para a leitura. Não é necessário imensas áreas em branco, mas agrupar pedaços de código através da utilização de espaços em branco funciona muito bem. Costuma-se separar parâmetros com espaços em branco.

Quebra de linhas muito longas

Com o objetivo de tornar o código mais fácil de ler e imprimir, as linhas do código não devem estender o limite da tela ou do papel. Podem ser "quebradas" em mais de uma linha de texto utilizando o ponto-e-vírgula (;).

Capitulação de palavras-chave

Uma convenção amplamente utilizada é a de capitular as palavras chaves, funções, variáveis e campos utilizando uma combinação de caracteres em maiúsculo e minúsculo, visando facilitar a leitura do código fonte.

Avaliando o código a seguir:

```
local ncnt
while ( ncnt++ < 10 )
ntotal += ncnt * 2
enddo
```

O mesmo ficaria muito mais claro se re-escrito conforme abaixo:

```
Local nCnt

While ( nCnt++ < 10 )
    nTotal += nCnt * 2
EndDo
```

Utilização da Notação Húngara

A Notação Húngara é muito comum entre programadores xBase e de outras linguagens. A documentação do ADVPL utiliza esta notação para a descrição das funções e comandos e é aconselhável sua utilização na criação de rotinas, pois ajuda a evitar pequenos erros e facilita a leitura do código. Para maiores detalhes, consulte a documentação sobre a utilização da Notação Húngara de códigos fontes disponível nos demais tópicos deste material.

Utilização de nomes significantes para variáveis

A principal vantagem da liberdade na criação dos nomes de variáveis é a facilidade de identificação da sua utilidade. Portanto deve-se utilizar essa facilidade o máximo possível. Nomes sem sentido apenas tornarão difícil a identificação da utilidade de uma determinada variável, assim como nomes extremamente curtos. Nem sempre a utilização de uma variável chamada *i* é a melhor saída. Claro, não convém criar uma variável com um nome muito longo que será utilizada como um contador, e referenciada muitas vezes no código. O bom senso deve ser utilizado.

Criar variáveis como *nNumero* ou *dData* também não ajudam na identificação. A Notação Húngara já está sendo utilizada para isso e o objetivo do nome da variável deveria ser identificar sua utilização, não o tipo de dado utilizado. Deve-se procurar substituir tais variáveis por algo como *nTotal* ou *dCompra*.

O mesmo é válido para nomes de funções, que devem descrever um pouco sobre o que a função faz.

Novamente nomes extremamente curtos não são aconselháveis.

Utilização de comentários

Comentários são muito úteis na documentação de programas criados e para facilitar a identificação de processos importantes no futuro e devem sempre ser utilizados.

Sempre que possível, funções criadas devem ter uma breve descrição do seu objetivo, parâmetros e retorno. Além de servir como documentação, os comentários embelezam o código ao separar as funções umas das outras. Os comentários devem ser utilizados com bom senso, pois reescrever a sintaxe ADVPL em português torna-se apenas perda de tempo:

```
If nLastKey == 27 // Se o nLastKey for igual a 27
```

Criação de mensagens sistêmicas significantes e consistentes

Seja oferecendo assistência, exibindo mensagens de aviso ou mantendo o usuário informado do estado de algum processo, as mensagens devem refletir o tom geral e a importância da aplicação. Em termos gerais, deve-se evitar ser muito informal e ao mesmo tempo muito técnico.

```
"Aguarde. Reindexando (FILIAL+COD+ LOCAL) do arquivo: \DADOSADV\SB1990.DBF"
```

Esse tipo de mensagem pode dar informações demais para o usuário e deixá-lo sentindo-se desconfortável se não souber o que significa "reindexando", etc. E de fato, o usuário não devia ser incomodado com tais detalhes. Apenas a frase "Aguarde, indexando." funcionaria corretamente, assim como palavras "processando" ou "reorganizando".

Outra boa idéia é evitar a referencia a um item corrente de uma tabela como um "registro":

```
"Deletar este registro?"
```

Se a operação estiver sendo efetuada em um arquivo de clientes, o usuário deve ser questionado sobre a remoção do cliente corrente, se possível informando valores de identificação como o código ou o nome.

Evitar abreviação de comandos em 4 letras

Apesar do ADVPL suportar a abreviação de comandos em quatro letras (por exemplo, repl no lugar de replace) não há necessidade de utilizar tal funcionalidade. Isto apenas torna o código mais difícil de ler e não torna a compilação mais rápida ou simples.

Evitar "disfarces" no código

Não deve-se criar constantes para expressões complexas. Isto tornará o código muito difícil de compreender e poderá causar erros primários, pois pode-se imaginar que uma atribuição é efetuada a uma variável quando na verdade há toda uma expressão disfarçada:

```
#define NUMLINES aPrintDefs[1]
#define NUMPAGES aPrintDefs[2]
#define ISDISK  aReturn[5]
```

```
If ISDISK == 1
    NUMLINES := 55
Endif
```

```
NUMPAGES += 1
```

A impressão que se tem após uma leitura deste código é de que valores estão sendo atribuídos às variáveis ou que constantes estão sendo utilizadas. Se o objetivo é flexibilidade, o código anterior deve ser substituído por:

```
#define NUMLINES 1
#define NUMPAGES 2
#define ISDISK  5

If aReturn[ISDISK] == 1
    aPrintDefs[ NUMLINES ] := 55
Endif
aPrintDefs[ NUMPAGES ] += 1
```

Evitar código de segurança desnecessário

Dada sua natureza binária, tudo pode ou não acontecer dentro de um computador. Adicionar pedaços de código apenas para "garantir a segurança" é freqüentemente utilizado como uma desculpa para evitar corrigir o problema real. Isto pode incluir a checagem para validar intervalos de datas ou para tipos de dados corretos, o que é comumente utilizando em funções:

```
Static Function MultMalor( nVal )
If ValType( nVal ) != "N"
    nVal := 0
Endif
Return ( nVal * nVal )
```

O ganho é irrisório na checagem do tipo de dado do parâmetro já que nenhum programa corretamente escrito em execução poderia enviar uma string ou uma data para a função. De fato, este tipo de "captura" é o que torna a depuração difícil, já que o retorno será sempre um valor válido (mesmo que o parâmetro recebido seja de tipo de dado incorreto). Se esta captura não tiver sido efetuada quando um possível erro de tipo de dado inválido ocorrer, o código pode ser corrigido para que este erro não mais aconteça.

Isolamento de strings de texto

No caso de mensagens e strings de texto, a centralização é um bom negócio. Pode-se colocar mensagens, caminhos para arquivos, e mesmo outros valores em um local específico. Isto os torna acessíveis de qualquer lugar no programa e fáceis de gerenciar.

Por exemplo, se existe uma mensagem comum como "Imprimindo, por favor aguarde..." em muitas partes do código, corre-se o risco de não seguir um padrão para uma das mensagens em algum lugar do código. E mantê-las em um único lugar, como um arquivo de cabeçalho, torna fácil a produção de documentação e a internacionalização em outros idiomas.