

# **ADVPL II**

Agosto/2010

---

**Sumário**

<b>1. PROGRAMANDO PARA BANCO DE DADOS.....</b>	<b>8</b>
<b>1.1. Diferenças e similaridades entre DBF e SQL.....</b>	<b>8</b>
1.2. Comandos e Funções para TopConnect .....	9
1.3. Exercícios SQL .....	13
1.4. Protheus Embedded SQL .....	14
1.5. Exercícios Embedded .....	15
<b>2. PROGRAMAÇÃO ORIENTADA A OBJETOS.....</b>	<b>16</b>
2.1. Conceitos Básicos.....	16
<b>3. CRIAÇÃO DE JANELAS PADRÃO OOP .....</b>	<b>20</b>
3.1. tWindow .....	19
3.2. MSDialog.....	22
<b>4. Desvendando as Classes .....</b>	<b>23</b>
4.1 Equivalência .....	23
4.2. tFont .....	26
4.3. tSay .....	28
4.4. tGet.....	<b>Erro! Indicador não definido.</b> 7
4.5. tButton .....	329
4.6. Exercício OO .....	31
<b>5. OBJETOS DE CONTROLE .....</b>	<b>32</b>
5.1. tControl .....	32
5.2. tButton .....	32
5.3. tCheckBox.....	34
5.4. tComboBox .....	407
5.5. Radio.....	429
<b>6. EDITORES DE TELA .....</b>	<b>41</b>
6.1. Exercício usando Editor de Tela .....	41
<b>7. MONTANDO JANELAS .....</b>	<b>43</b>

---

---

7.1. ListBox .....	43
7.2. Exercício ListBox .....	51
7.3. DBTree .....	52
7.4. Enchoice .....	50
7.5. MSGetDados.....	556
7.6. MSNewGetDados.....	57
7.7. MSSelect.....	61
7.8. MSGetdb .....	67
<b>8. RELATÓRIOS GRÁFICOS.....</b>	<b>68</b>
8.1. TMSPrinter .....	68
8.2. Exercício TMSPrinter .....	71
8.3. TReport.....	72
<b>9. PACOTE OFFICE .....</b>	<b>79</b>
9.1. Integração Excel .....	79
9.2. Exercício Excel .....	83
9.3. Integração com Word .....	85

---

## Capítulo 01 – PROGRAMANDO PARA BANCO DE DADOS

O que você aprenderá neste capítulo: Podemos utilizar queries no Protheus quando acessamos bancos de dados via TOPCONNECT.

A utilização de queries melhoram consideravelmente a velocidade das consultas aos dados e reduzem a sobrecarga no servidor de aplicação.

## Capítulo 02 – PROGRAMAÇÃO ORIENTADA A OBJETOS

O que você aprenderá neste capítulo: A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema de *software*.

## Capítulo 03 – CRIAÇÃO DE JANELAS PADRÃO OOP

O que você aprenderá neste capítulo: Criar janelas e customizar janelas de interface com o usuário.

## Capítulo 04 – Desvendando as Classes

O que você aprenderá neste capítulo: Criar e manipular as classes, seus métodos e atributos.

## Capítulo 05 – OBJETOS DE CONTROLE

O que você aprenderá neste capítulo: Classe abstrata mãe de todos os controles editáveis. TCONTROL é a classe comum entre todos os componentes visuais editáveis.

## Capítulo 06 – EDITORES DE TELA

O que você aprenderá neste capítulo: Existem diversos editores visuais de tela.

GAIA – Desenvolvido pela Microsiga

MKW - Nativo do Clipper

Designer - Desenvolvido por Analista de Mercado.

**OBS.** Todos geram o código em ADVPL.

---

## Capítulo 07 – MONTANDO JANELAS

O que você aprenderá neste capítulo: O conteúdo deste capítulo abrange um pouco mais sobre janelas e seus componentes.

## Capítulo 08 – RELATÓRIOS GRÁFICOS

O que você aprenderá neste capítulo: Criar e customizar relatórios.

## Capítulo 09 – PACOTE OFFICE

O que você aprenderá neste capítulo: Integrar o Protheus com as ferramentas do Microsoft Office.

Podemos utilizar queries no Protheus quando acessamos bancos de dados via TOPCONNECT.

A utilização de queries melhoram consideravelmente a velocidade das consultas aos dados e reduzem a sobrecarga no servidor de aplicação, TopConnect e Banco de Dados.

Normalmente uma query é utilizada em substituição a um Loop (While) na base de dados de programação convencional. Queries mais complexas utilizando joins poder ser construídas com a mesma função de vários loops.

### **Diferenças e similaridades entre DBF e SQL**

A princípio não existem diferenças na programação para a versão SQL, já que pelo próprio fato de ser uma linguagem interpretada, o sistema é quem se encarrega de executar os comandos e funções adequadamente no ambiente em que trabalha. Mas é importante manter algumas informações em mente ao programar para o ambiente SQL.

Deve-se lembrar que estamos trabalhando com um banco de dados relacional, que se utiliza de tabelas ao invés de arquivos, e onde o sistema não tem acesso aos dados de forma nativa e sim através do TOPCONNECT. Essa forma de acesso adiciona ao sistema algumas das características e vantagens oferecidas pelo SGBD em uso (por exemplo, o Oracle, MSSQL Server ou o DB2) como, por exemplo, segurança e integridade referencial, e as imensas facilidades da linguagem SQL, mas por outro lado tem-se também as implicações da conversão dos comandos no padrão XBASE para a perfeita compreensão no ambiente SQL.

Imagine a montagem de uma expressão de filtro para um índice condicional. Tome a seguinte expressão como exemplo: "DTOS(E1\_VENCTO) >= DTOS(mv\_par01)". Em um ambiente padrão XBASE, como o NTX ou o ADS, pode-se utilizar variáveis sem qualquer problema em uma expressão de filtro pois a mesma será avaliada registro a registro durante a montagem do índice. Mas no ambiente SQL, o filtro nada mais é do que uma tabela temporária, onde estão selecionados apenas os registros conforme a condição indicada. A seleção de dados em tabelas pelo SQL é mais rápida, mas em compensação o SGBD não tem como reconhecer a variável informada na expressão. Ela existe apenas no sistema ou, mais especificamente, no seu programa.

Por isso, deve-se substituir a expressão anteriormente exemplificada pela seguinte (que também funcionaria perfeitamente em um ambiente XBASE): "DTOS(E1\_VENCTO) >= '"+DTOS(mv\_par01)+'"'. Esta expressão é melhor que anterior simplesmente porque não se utiliza variável e sim do conteúdo da mesma, o que pode ser compreendido em qualquer ambiente. Todas essas explicações são válidas, da mesma maneira, a filtros criados através do comando SET FILTER.

Ainda existem outros detalhes a se considerar quando se trabalha com índices em um ambiente SQL. É que na verdade não existem índices condicionais nesse ambiente. O filtro é criado independente do índice. Então, você pode criar um INDREGUA com um filtro e mudar a ordem, mas o filtro permanecerá ativo, em qualquer ordem. Do mesmo modo, não

se podem manter, dois índices, com filtros diferentes, pois um filtro sobrescreveria o outro.

Outro ponto de atenção deve ser a função XBASE chamada DBSETINDEX. Podem ocorrer alguns erros ao tentar-se utilizar essa função para abrir um índice de trabalho criado. Por esses motivos e pelo fato de tornar o processamento mais lento deve-se evitar ao máximo o uso de índices de trabalho no ambiente SQL.

Da mesma maneira que a função DBSETINDEX, os comandos COPY TO e APPEND FROM também devem ter uma atenção especial. No ambiente SQL esses comandos são executados entre uma tabela e um arquivo DBF (e vice-versa) ou entre dois arquivos DBF. Por exemplo, o comando COPY TO pode ser usado para copiar os dados da tabela ativa para um DBF local e o comando APPEND FROM pode ser usado para importar os dados de um arquivo local para a tabela ativa. Os dois podem ser usados entre dois arquivos, mas nunca se pode usar, por exemplo, o comando APPEND FROM para importar os dados de uma tabela para outra.

### **Comandos e Funções para TOPCONNECT**

**TCCONTYPE** - Define o tipo de conexão que será utilizada entre o Protheus e o TOPCONNECT.

**Sintaxe:** TCCONTYPE (cTipo)

Argumento	Obrigat.	Tipo	Descrição
cTipo	Sim	C	Tipo da conexão. Pode ser: "TCPIP" ou "NPIPE"

Exemplo:

```
TCConType("NPIPE")  
TCConType("TCPIP")
```

**TCDELFILE** - Apaga um arquivo de um banco de dados.

**Sintaxe:** TCDELFILE (cTabela)

Argumento	Obrigat.	Tipo	Descrição
cTabela	Sim	C	Nome da tabela que deve ser apagada.

Exemplo:

```
If TcDelFile("SA1020")
```

```
MSGINFO("Tabela excluída com sucesso")
Else
MSGINFO("Não foi possível excluir a tabela")
Endif
```

**TCGENQRY** - Define a execução de uma Query, a próxima chamada à DBUSEAREA será a abertura de uma Query e não de tabela.

**Sintaxe:** TCGENQRY ([XPAR1, XPAR2,], CQUERY)

Argumento	Obrigat.	Tipo	Descrição
<i>xPar1</i> , <i>xPar2</i>	Não	Qualquer	Parâmetros apenas para compatibilização. Não tem função.
cQuery	Sim	C	Contém a expressão da query que se deseja executar.

Exemplo:

```
cQuery := 'SELECT X2_CHAVE CHAVE, R_E_C_N_O_ RECNO from SX2990'
DbUseArea (.T., 'TOPCONN', TCGenQry(,cQuery), 'TRB', .F., .T.)
```

**TCSETFIELD** - Define formato de campos numericos e data de acordo com o dicionário.

**Sintaxe:** TCSETFIELD (cAlias cCampo, cTipo, nTam, nDec)

Argumento	Obrigat.	Tipo	Descrição
<i>cAlias</i>	Sim	C	Alias para aplicação o formato
cCampo	Sim	C	Nome do campo a formatar
cTipo	Sim	C	Tipo do campo
nTam	Sim	N	Tamanho
nDec	Sim	N	Decimal

Exemplo:

```
cQuery := 'SELECT E2_NUM, E2_VALOR, E2_EMISSAO from SE2990'
dbUseArea(.T., 'TOPCONN', TCGenQry(,cQuery), 'TRB', .F., .T.)
TCSetField( "TRB", "E2_VALOR", "N", 17, 2 )
TCSetField( "TRB", "E2_EMISSAO", "D", 8, 0 )
```



**RETSQLNAME** - Retorna o nome da tabela de acordo com a empresa posicionada.

**Sintaxe:** RETSQLNAME (cAlias)

Argumento	Obrigat.	Tipo	Descrição
CAlias	Sim	C	Alias desejado

Exemplo:

```
cQry1 := "SELECT * FROM " + RetSqlName("SZ2") + " WHERE D_E_L_E_T_ = 
""")
```

**SQLORDER** - Retorna a instrução do índice corrente.

**Sintaxe:** SQLORDER (cAlias)

Argumento	Obrigat.	Tipo	Descrição
cAlias	Sim	N ou C	Instrução com a chave, se numérico, busca a expressão da ordem informada, se caracter utiliza a própria expressão informada.

Exemplo:

```
cQry1 := "SELECT * FROM " + RetSqlName("SZ2") + " WHERE D_E_L_E_T_ = 
""")
cQry1+= "ORDER BY " + SqlOrder("Z2_FILIAL+Z2_CODIGO")
ou
cQry1 := "SELECT * FROM " + RetSqlName("SZ2") + " WHERE D_E_L_E_T_ = 
""")
cQry1+= "ORDER BY " + SqlOrder(SZ2->(indexkey()))
ou
cQry1 := "SELECT * FROM " + RetSqlName("SZ2") + " WHERE D_E_L_E_T_ = 
""")
cQry1+= "ORDER BY " + SqlOrder(SZ2->(indexorder()))
```

**TCREFRESH** - Faz refresh em uma tabela, através de uma leitura forçada da tabela no banco de dados. Utilizada após o DELETE e o INSERT.

**Sintaxe:** TCREFRESH ( cTabela )

Argument	Obrigat.	Tipo	Descrição
----------	----------	------	-----------

<b>o</b>			
cTabela	Sim	Lógico	Indica nome da tabela que deve ser feito refresh.

Exemplo:

```
cTabela:= "SA1990"
cComando := "Delete " + cTabela + " Where R_E_C_N_O_ > 50000 "
TCSqlExec(cComando)
TCRefresh(cTabela)
```

**TCSQLEXEC** - Executa um comando em SQL

**Sintaxe:** TCSQLEXEC (cComando)

Argumento	Obrigat.	Tipo	Descrição
cComando	Sim	C	Instrução em SQL.

Exemplo:

```
TCSQLEXEC("DELETE " + RetSqlName("SZ2") + " WHERE D_E_L_E_T_ = '*')"
```

**TCSPEXIST** - Verifica se uma Stored Procedure existe.

**Sintaxe:** TCSPEXIST( cStoredProc )

Argumento	Obrigat.	Tipo	Descrição
CStoredProc	Sim	C	Nome da Stored Procedure.

Exemplo:

```
if TCSPEXist("SP000001")
    cStr := "DROP PROCEDURE " + "SP000001 "
    TCSqlExec(cStr)
endif
```

**TCSPEXEC** - Executa Executa uma Stored Procedure, no banco de dados, com número variável de parâmetros. Retorna um array contendo os valores de retorno da SP

Sintaxe

TCSPEXEC (cSProc [, xParam1,...,xParamN])--> [array]

Argumento	Obrigat.	Tipo	Descrição
cSProc	Sim	C	Nome da Stored Procedure.
xParamX	Não	Qualquer	Parâmetro(s) da Stored Procedure

Exemplo:

```
cQryProc := "Create Procedure teste1( @IN_VALUE int, @OUT_STR char(255),
@OUT_VALUE int) " +CRLF
cQryProc += "WITH RECOMPILE" +CRLF
cQryProc += "As" +CRLF
cQryProc += "Begin" +CRLF
cQryProc += "  Select @OUT_STR = "Teste", @OUT_VALUE = @IN_VALUE + 3"
+CRLF
cQryProc += "End" +CRLF
cQryProc += "GO" +CRLF

aResult := TCSPEXEC(xProcedures ('teste1'), 100 )
```

### **Exercícios SQL**

1. Fazer um programa utilizando funções de bancos de Dados que imprima os seguintes campos:

#### **Campos:**

No. Pedido, Nome do Cliente, Codigo do Produto, Descrição do Produto, Data de Emissao do Pedido.

#### **Tabelas de Apoio: SC5, SC6, SA1, SB1**

Fazer um programa utilizando funções de bancos de Dados.

#### **Parametros:**

Cliente Inicial (SA1)  
 Cliente Final (SA1)  
 Ordena por: Cliente/Produto  
 Analitico / Sintético

#### **Campos:**

Nome do Cliente, Numero da Nota, Data de Emissao, Código do Produto, Descrição do Produto, Quantidade, Valor Unitário, Total

## Tabelas de Apoio: SD2, SF2, SA1, SB1

### Protheus Embedded SQL

O objetivo do *Embedded SQL* é facilitar a escrita e a leitura de consultas ao banco de dados. Para isso, foi definida uma sintaxe para que se possa escrever a consulta diretamente no código *AdvPL*, sem ter que montar códigos complexos que dificultam a leitura do comando, porém sua utilização está disponível apenas para build 7.00.050721p ou superior.

Instrução	Descrição
BeginSQL Alias	Inicia o Embedded SQL
EndSQL	Finaliza a instrução
%xxx%	Instrução literal a ser substituída
Column	Especifica tipo de conversão para campos dos tipos data, numérico ou lógico utilizado em substituição do TCSetField
%exp.%	Utilizado para especificar variáveis, expressões e funções.
%NoParcer%	indica que a consulta não deve passar pela função <b>"ChangeQuery"</b>
%Table: <cAlias> %	Indica a tabela a ser utilizada, substitui a função <b>"RetSqlName"</b>
%NotDel%	Substitui a expressão <code>D_E_L_E_T_ = " "</code>
%Order: <alias> %	Define a ordem a utilizar no ORDER BY, substitui a função <b>SqlOrder(&lt;Alias&gt;-&gt;(IndexKey()))</b>
%Order: <cAlias> , <nIndice> %	Define a ordem a utilizar no ORDER BY, substitui a função <b>SqlOrder(&lt;Alias&gt;-&gt;(Indexorder(nIndice)))</b>
%Order: <cAlias> , <nNick> %	Define a ordem a utilizar no ORDER BY, substitui a função <b>SqlOrder(&lt;Alias&gt;-&gt;(DBNickIndexKey(cNick)))</b>
<b>GetLastQuery()</b>	Retorna um vetor com cinco elementos, com as seguintes informações da ultima query executada: [1] - Alias usado para abrir o cursor [2] - Query executada

	<p>[3] –Vetor de campos com critérios de conversão especificados</p> <p>[4] –Caso .T. não foi utilizada QueryChange na string original</p> <p>[5] –Tempo em segundos utilizado para a abertura do cursor</p>
--	--

## Limitações:

A utilização de funções dentro do Código Embedded não é permitida, caso necessite execute a função antes de iniciar o embedded pois a instrução só recebe variáveis resolvidas.

Não é possível debugar.

Exemplo:

### Utilizando a forma usual

```

cADQry := "SELECT C6_NUM, C6_PRODUTO, C6_QTDVEN, C6_PRCVEN ,
C6_ENTREG" + CRLF
cADQry += "FROM " + RetSqlName("SC6") + " C6 "+CRLF
cADQry += "WHERE C6_FILIAL = '" + xFilial("SC6") + "' AND D_E_L_E_T_ =
'' "+CRLF
cADQry += "ORDER BY "+RetSqlOrder("SC6") +CRLF
dbUseArea( .T., "TOPCONN", TcGenQry(,cQuery),"TRB", .T., .T. )
TcSetField("TRB", "C6_QTDVEN", "N", 17, 5 )
TcSetField("TRB", "C6_PRCVEN", "N", 17, 5 )
TcSetField("TRB", "C6_ENTREG", "D", 17, 5 )

```

### Utilizando *Embedded SQL*

```

BeginSQL Alias "TRB"
    Column C6_QTDVEN as numeric(17,2)
    Column C6_PRCVEN as numeric(17,2)
    Column C6_ENTREG as date
    %NoParser%
    SELECT C6_NUM, C6_PRODUTO, C6_QTDVEN, C6_PRCVEN,
C6_ENTREG
    FROM %Table:SC6% C6
    WHERE C6_FILIAL = %xFilial:SC6% AND %NotDel%

```

```
ORDER BY %Order:SC6%  
EndSQL
```

### **Exercícios Embedded**

1. Fazer um programa utilizando Embedded SQL que imprima os seguintes campos:

**Campos:**

No. Pedido, Nome do Cliente, Código do Produto, Descrição do Produto, Data de Emissão do Pedido.

**Tabelas de Apoio: SC5, SC6, SA1, SB1**

## Programação Orientada a Objetos

A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema de *software*. O funcionamento deste sistema se dá através do relacionamento e troca de mensagens entre estes objetos.

Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de *software*. Cada classe determina o comportamento (definidos nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos.

Programas que utilizam conceitos OO, ao invés de definir funções independentes que são utilizadas em conjunto, dividem conceitualmente o “problema” em partes independentes (objetos), que podem conter atributos que os descrevem, e que implementam o comportamento do sistema através de funções definidas nestes objetos (métodos). Objetos (e seus métodos) fazem referência a outros objetos e métodos; o termo “envio de mensagens” é utilizado para descrever a comunicação que ocorre entre os métodos dos diferentes objetos.

Na prática, um programa orientado a objetos pode ser descrito como um conjunto de classes pré-definidas ou definidas pelo usuário que possuem atributos e métodos, e que são instanciadas em objetos, durante a execução do programa.

### Conceitos Básicos

**Classe** representa um conjunto de objetos com diversas características. Uma classe define o comportamento dos objetos, através de métodos, e quais estados ele é capaz de manter, através de atributos.

Exemplo de classe: Empresa.

**Objeto** é uma instância de uma classe. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos.

Exemplo de objetos da classe Empresa: Microsiga, ADVPL, Biale.

**Atributos** são os dados ou informações do objeto, basicamente a estrutura de dados que vai representar a classe.

Exemplos:

Funcionário: nome, endereço, telefone

Cursos: nome, Tempo, preço

Alunos: Nome, endereço, telefone

**Métodos** definem as habilidades de cada objeto.

Biale é uma instância da classe Empresa(objeto), que tem habilidade para dar suporte, implementada através do método SuporteCliente(). A ação só ocorre quando o método é invocado através do objeto, no caso Biale. Dentro do programa, a utilização de um método deve afetar apenas um objeto em particular; Todas as empresas podem dar suporte ao cliente, mas você quer que apenas a Biale de o suporte. Normalmente, uma classe possui diversos métodos, que no caso da classe empresa poderiam ser treina(), atendetelefone() .

**Mensagem** é uma chamada a um objeto para invocar um de seus métodos, ativando um comportamento descrito por sua classe. Também pode ser direcionada diretamente a uma classe.

**Sobrecarga** é a utilização do mesmo nome para símbolos ou métodos com operações ou funcionalidades distintas. Geralmente diferencia-se os métodos pela sua assinatura.

**Herança** é o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e estados possíveis (atributos). Há Herança múltipla quando uma sub-classe possui mais de uma super-classe. Essa relação é normalmente chamada de relação "é um".

Um exemplo de herança: Filial é super-classe de Empresa. Ou seja, uma Filial é uma empresa.

**Associação** é o mecanismo pelo qual um objeto utiliza os recursos de outro. Pode tratar-se de uma associação simples "usa um" ou de um acoplamento "parte de".

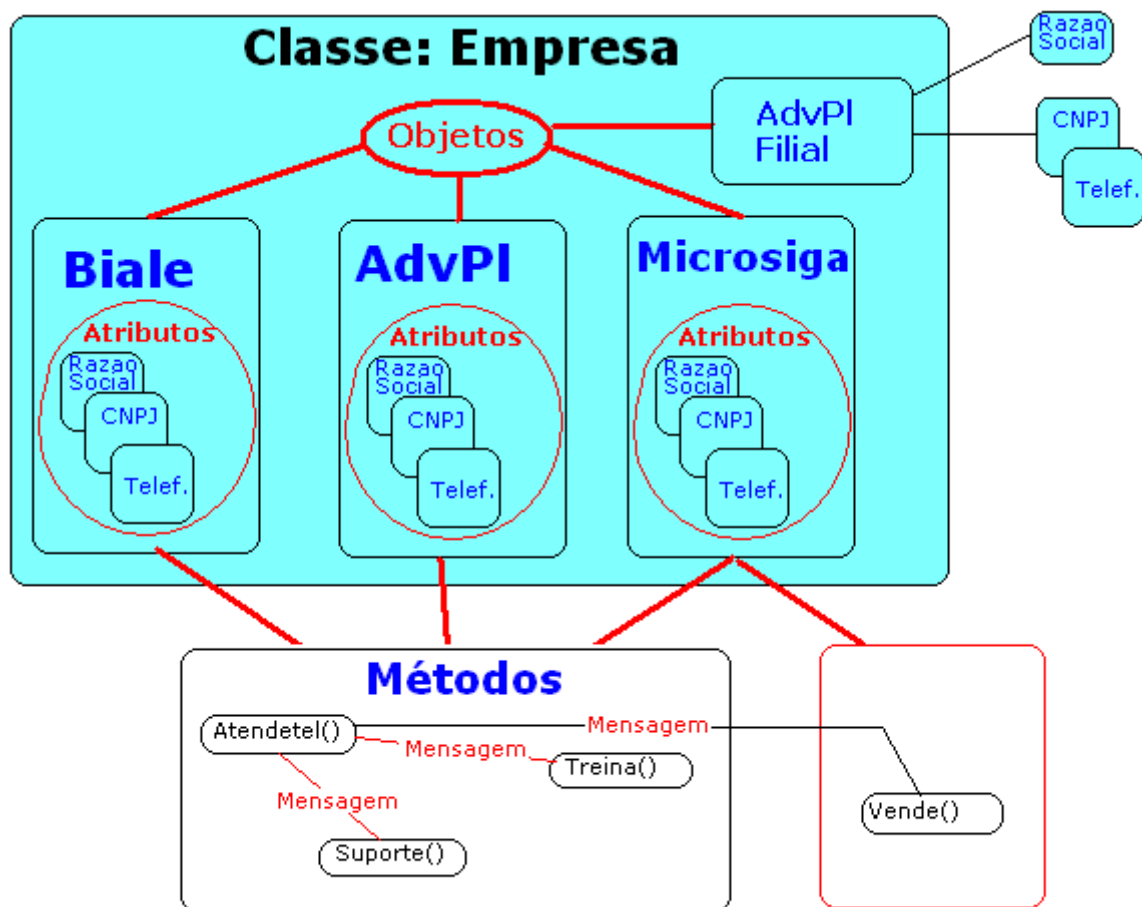
Por exemplo: Uma empresa usa um telefone. A tecla "1" é parte de um telefone.

**Encapsulamento** consiste na separação de aspectos internos e externos de um objeto. Este mecanismo é utilizado para impedir o acesso direto aos atributos de um objeto, disponibilizando externamente apenas os métodos que alteram estes estados. Exemplo: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões, o monofone e os sinais de tom).

**Polimorfismo** é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura (lista de parâmetros e retorno) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse. A decisão sobre qual o método que deve ser selecionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução. No caso de polimorfismo, é necessário que os métodos tenham exatamente a mesma identificação, sendo utilizado o mecanismo de redefinição de métodos.

**Exemplo Ilustrativo de Classe**





## CRIAÇÃO DE JANELAS PADRÃO OOP

### TWINDOW

Classe de janela principal de programa, deverá existir apenas uma instância deste objeto na execução do programa.

#### Propriedades

bInit	Bloco de código. Executado quando a janela está sendo exibida.
IEscClose	Lógico. Se .T. habilita o <ESC> cancelar a execução da janela.
oCtlFocus	Objeto. Objeto contido na janela que está com foco de entrada de dados

#### Métodos

**New** - Método construtor da janela.

#### Sintaxe

New( [anTop], [anLeft],[anBottom], [anRight], [acTitle], [nPar6], [oPar7], [oPar8],[oPar9], [aoParent], [IPar11],;[IPar12], [anClrFore], [anClrBack], [oPar15], [cPar16], [IPar17], [IPar18], [IPar19], [IPar20],[alPixel] )

Argumento	Obrigat.	Tipo	Descrição
AnTop	Não	N	Coordenada vertical superior em pixels ou caracteres.
AnLeft	Não	N	Coordenada horizontal esquerda em pixels ou caracteres.
AnBottom	Não	N	Coordenada vertical inferior em pixels ou caracteres.
AnRight	Não	N	Coordenada horizontal inferior em pixels ou caracteres.
acTitle	Não	C	Título da janela.
nPar6			Reservado.
oPar7			Reservado.
oPar8			Reservado.
oPar9			Reservado.
AoParent	Não	Obj	Janela mãe da janela corrente.

IPar11			Reservado.
IPar12			Reservado.
anClrFor e	Não	N	Cor de fundo da janela.
anClrTex t	Não	N	Cor do texto da janela.
oPar15			Reservado.
cPar16			Reservado.
IPar17			Reservado.
IPar18			Reservado.
IPar19			Reservado.
IPar20			Reservado.
AlPixel	Não	Log	Se .T. (padrão) considera coordenadas passadas em pixels, se .F. considera caracteres.

**Activate** - Ativa (exibe) a janela. Chamar esse método apenas uma vez.

### Sintaxe

Activate([acShow], [bPar2], [bPar3], [bPar4], [bPar5], [bPar6], [ abInit ], [bPar8], [bPar9], [bPar10],;

bPar11],[bPar12] ,[bPar13], [bPar14], [bPar15], [abValid], [bPar17], [bPar18] )

Argumento	Obrigat.	Tipo	Descrição
AcShow	Não	C	"ICONIZED" para janela iconizada ou "MAXIMIZED" para janela maximizada.
bPar2			Reservado.
bPar3			Reservado.
bPar4			Reservado.
bPar5			Reservado.
bPar6			Reservado.

AbInit	Sim	Bloco	Executado quando janela está sendo exibida.
bPar8			Reservado.
bPar9			Reservado.
bPar10			Reservado.
bPar11			Reservado.
bPar12			Reservado.
bPar13			Reservado.
bPar14			Reservado.
bPar15			Reservado.
AbValid	Sim	Bloco	Bloco de código. Executado quando a janela for solicitada de fechar. Deverá retornar .T. se o conteúdo da janela for válido, ou .F. se não. Se o bloco retornar .F. a janela não fechará.
bPar17			Reservado.
bPar18			Reservado.

**End** – Solicita fechamento da janela.

**Sintaxe:** End()

**Retorno** - Lógico. .T. se encerrou a janela e .F. se não.

**Center** – Centraliza a janela.

**Sintaxe:** Center()

Exemplo:

```
#INCLUDE "PROTHEUS.CH"
USER FUNCTION Teste()
Local oWindow
Local abInit:= {||conout("ativando!")}
Local abValid:= {||conout("encerrando!")}.T.}
oWindow:= tWindow():New( 10, 10, 200, 200, "Meu programa",,,,,,
CLR_WHITE,;
CLR_BLACK,,,,,,.T. )
```

```
oWindow:Activate("MAXIMIZED",,,,,,abInit,,,,,,,,,abValid,,)
/* os comandos abaixo proporcionam o mesmo resultado
  DEFINE WINDOW oWindow FROM 10, 10 TO 200,200 PIXEL TITLE
  "Meu programa" COLOR CLR_WHITE,CLR_BLACK
  ACTIVATE WINDOW oWindow MAXIMIZED ON INIT abInit VALID
  abValid
*/
Return NIL
```

## **MSDialog**

MSDialog 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.

### **Métodos**

**New** - Método construtor da classe.

**Sintaxe:** New([anTop], [anLeft], [anBottom], [anRight], [acCaption], [cPar6], [nPar7], [lPar8], [nStyle], [anClrText], [anClrBack], [oBrush], [aoWnd], [alPixel], [oIco], [oFont])

Argumento	Obrigatório	Tipo	Descrição
anTop	Não	N	Coordenada vertical superior em pixels ou caracteres.
anLeft	Não	N	Coordenada horizontal esquerda em pixels ou caracteres.
anBottom	Não	N	Coordenada vertical inferior em pixels ou caracteres.
anRight	Não	N	Coordenada horizontal direita em pixels ou caracteres.
acCaption	Não	C	Título da janela.
cPar6..cPar9			Reservado.
nStyle	Não	N	Estilo da Janela
anClrText	Não	N	Cor do texto.
anClrBack	Não	N	Cor de fundo.
oBrush	Não	Obj	Nome do objeto com a classe Brush
aoWnd	Não	Obj	Janela mãe da janela a ser criada, padrão é a janela principal do programa.
alPixel	Não	Lóg	Se .T. considera as coordenadas passadas em pixels, se .F. considera caracteres.
oIco	Não	Obj	Objeto do Icone
oFont	Não	Obj	Objeto da fonte da janela, se nulo, Arial 10

Exemplo:

```
#INCLUDE "protheus.ch"
User Function Teste()

// cria diálogo
Local oDlg:=MSDialog():New(10,10,300,300,"Meu
diálogo",,,,,CLR_BLACK,CLR_WHITE,,,T.)

// ativa diálogo centralizado
oDlg:Activate(,,,T.,{||msgstop("validou!"),.T.},{||msgstop("iniciando...") }
Return
```

## Desvendando as classes

### Equivalência

A utilização dos includes nos permite a utilização dos objetos a partir de equivalências, veja a seguir:

```
#xcommand DEFINE MSDIALOG <oDlg> ;
[ <resource: NAME, RESNAME, RESOURCE> <cResName> ] ;
[ TITLE <cTitle> ] ;
[ FROM <nTop>, <nLeft> TO <nBottom>, <nRight> ] ;
[ <lib: LIBRARY, DLL> <hResources> ] ;
[ <vbx: VBX> ] ;
[ STYLE <nStyle> ] ;
[ <color: COLOR, COLORS> <nClrText> [,<nClrBack> ] ] ;
[ BRUSH <oBrush> ] ;
[ <of: WINDOW, DIALOG, OF> <oWnd> ] ;
[ <pixel: PIXEL> ] ;
[ ICON <oIco> ] ;
[ FONT <oFont> ] ;
[ <status: STATUS> ] ;
=> ; //Equivalencia
```

```
<oDlg> = MsDialog():New(<nTop>, <nLeft>, <nBottom>, <nRight>, <cTitle>,
<cResName>, <hResources>,,;
```

```
<.vbx.>, <nStyle>, <nClrText>, <nClrBack>, <oBrush>, <oWnd>, <.pixel.>;,  
<oIco>, <oFont> , <.status.> )
```

Exemplo utilizando forma equivalente a Classe MSDialog:

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

```
User Function fTela1()
```

```
Local nVar := 0
```

```
Private oDlg
```

```
Define MSDialog oDlg Title OemToAnsi("Titulo da janela") From 0,0 To 160,380  
Pixel
```

```
@05,10 To 50,180 Pixel
```

```
@15,20 Say "Colocar aqui a mensagem que quiser" Pixel Of oDlg
```

```
@25,20 MSGet oVar Var nVar Picture "@E 999,999.99" Size 50,10 Pixel Of oDlg
```

```
@70,20 Button oBtnOk Prompt "&Ok" Size 30,15 Pixel Action Fecha(oDlg) Of oDlg
```

```
@70,80 Button oBtnCancel Prompt "&Cancelar" Size 30,15 Pixel ;
```

```
        Action (msginfo("Cliquei no Cancelar"), oDlg:End()) Cancel Of oDlg
```

```
Activate MSDialog oDlg Centered
```

```
Static Function Fecha(oDlg)
```

```
    (msginfo("Cliquei no OK"))
```

```
    oDlg:End()
```

```
Return
```

## tFont

Classe que encapsula fonte de edição.

**Métodos** - Método construtor da classe.

**New** -

**Sintaxe:** New([acName], [nPar2], [anHeight], [lPar4], [alBold], [nPar6], [lPar7],;  
[nPar8], [alItalic], [alUnderline])

Argumento	Obrigat.	Tipo	Descrição
acName	Não	C	Nome da fonte, o padrão é "Arial".
nPar2			Reservado.
anHeight	Não	N	Tamanho da fonte. O padrão é -11.



IPar4			Reservado.
AlBold	Não	Log	Se .T. o estilo da fonte será negrito.
nPar6			Reservado.
IPar7			Reservado.
nPar8			Reservado.
alItalic	Não	Log	Se .T. o estilo da fonte será itálico.
alUnderline	Não	Log	Se .T. o estilo da fonte será sublinhado.

#### Exemplo

```
#include "protheus.ch"
User Function Teste()

Local oDlg, oSay
Local oFont:= TFont():New("Courier New",-14,.T.)

DEFINE MSDIALOG oDlg FROM 0,0 TO 200,200 TITLE "Minha tela com Fonte Courier
New" PIXEL
    // apresenta o tSay com a fonte Courier New //
    oSay:= tSay():New(10,10,{||"para
exibir"},oDlg,,oFont,,,,.T.,CLR_WHITE,CLR_RED,100,20)
    ACTIVATE MSDIALOG oDlg CENTERED
Return
```

#### Exemplo utilizando forma equivalente a Classe TFont

```
#include "protheus.ch"
User Function Telafont()
Local oDlg, oSay, oFont
DEFINE FONT oFont Name "Courier New" SIZE 0,-14 BOLD

DEFINE MSDIALOG oDlg FROM 0,0 TO 200,200 TITLE "Minha tela Courier New" PIXEL
    @ 010,010 SAY "para exibir" SIZE 100,20 FONT oFont COLOR CLR_RED PIXEL
of ODlg
    ACTIVATE MSDIALOG oDlg CENTERED
Return
```

## TSAY

Classe de objetos visuais que exibe o conteúdo de texto estático sobre uma janela ou controle previamente definidos.

Propriedades:

<b>IWordWrap</b>	Lógico. Se .T. quebra o texto em várias linhas de maneira a enquadrar o conteúdo na área determinada para o controle, sendo o padrão .F.
<b>ITransparent</b>	Lógico. Se .T. a cor de fundo do controle é ignorada assumindo o conteúdo ou cor do controle ou janela ao fundo, sendo o padrão .T.

### Métodos

**New** - Método construtor da classe.

**Sintaxe** New ([anRow], [anCol], [abText], [aoWnd], [acPicture], [aoFont], [IPar7], [IPar8], [IPar9], [alPixels], [anClrText], [anClrBack], [anWidth], [anHeight])

Argumento	Obrigat.	Tipo	Descrição
anRow	Sim	N	Coordenada vertical em pixels ou caracteres.
anCol	Sim	N	Coordenada horizontal em pixels ou caracteres.
abText	Sim	B	Quando executado deve retornar uma cadeia de caracteres a ser exibida.
aoWnd	Sim	O	Janela ou diálogo onde o controle será criado.
acPicture	Não	C	Picture de formatação do conteúdo a ser exibido.
aoFont	Não	O	Objeto tipo tFont para configuração do tipo de fonte que será utilizado para exibir o conteúdo.
IPar7	Não	L	Reservado.
IPar8	Não	L	Reservado.
IPar9	Não	L	Reservado.
alPixels	Não	L	Se .T. considera coordenadas passadas em pixels se .F., padrão, considera as coordenadas passadas em caracteres.
anClrText	Não	N	Cor do conteúdo do controle.

anClrBack	Não	N	Cor do fundo do controle.
anWidth	Não	N	Largura do controle em pixels.
anHeight	Não	N	Altura do controle em pixels.

## Exemplo

```
#include "protheus.ch"
User Function Teste()
Local oDlg, oSay
Local oFont:= TFont():New("Courier New",-14,.T.)

DEFINE MSDIALOG oDlg FROM 0,0 TO 200,200 TITLE "Minha tela com Fonte Courier
New" PIXEL
    // apresenta o tSay com a fonte Courier New //
    oSay:= tSay():New(10,10,{||"para
exibir"},oDlg,,oFont,,,,.T.,CLR_WHITE,CLR_RED,100,20)
    ACTIVATE MSDIALOG oDlg CENTERED
```

Return

## Exemplo utilizando forma equivalente a Classe TSay

```
#include "protheus.ch"
User Function Telafont()
Local oDlg, oSay, oFont
DEFINE FONT oFont Name "Courier New" SIZE 0,-14 BOLD

DEFINE MSDIALOG oDlg FROM 0,0 TO 200,200 TITLE "Minha tela Courier New" PIXEL
@ 010,010 SAY "para exibir" SIZE 100,20 FONT oFont COLOR CLR_RED PIXEL of
ODlg
    ACTIVATE MSDIALOG oDlg CENTERED
```

Return

## TGET

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.

## Propriedades:

IPassword	Lógico. Se .T. o controle se comporta como entrada de dados de senha, exibindo asteriscos '*' para esconder o conteúdo digitado.
Picture	Caractere. Máscara de formatação do conteúdo a ser exibido.

## Métodos

**New** - Método construtor da classe.

**Sintaxe** 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])

Argumento	Obrigat.	Tipo	Descrição
AnRow	Sim	N	Coordenada vertical em pixels ou caracteres.
AnCol	Sim	N	Coordenada horizontal em pixels ou caracteres.
AbSetGet	Sim	B	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	Sim	O	Janela ou controle onde o controle será criado.
anWidth	Não	N	Largura do controle em pixels.
anHeight	Não	N	Altura do controle em pixels.
acPict	Não	C	Máscara de formatação do conteúdo a ser exibido.
abValid	Não	B	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	Não	N	Cor de fundo do controle.
anClrBack	Não	N	Cor do texto do controle.
aoFont	Não	O	Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do Controle
IPar12	Não	L	Reservado
oPar13	Não	O	Reservado.
alPixel	Não	L	Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
cPar15	Não	C	Reservado.
IPar16	Não	L	Reservado.

abWhen	Não	B	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.
IPar18	Não	L	Reservado.
IPar19	Não	L	Reservado.
abChange	Não	B	Executado quando o controle modifica o valor da variável associada.
alReadOnly	Não	L	Se .T. o controle não poderá ser editado.
alPassword	Não	L	Se .T. o controle exibirá asteriscos "*" no lugar dos caracteres exibidos pelo controle para simular entrada de senha.
cPar23	Não	C	Reservado.
acReadVar	Não	C	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( ).

## Exemplo

```
#include 'protheus.ch'
User Function TesteGet()
Local oDlg, oButton, oCombo, cCombo, nGet1:=0
    DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu Get'
        oGet1:= TGet():New(10,10,, oDlg,100,20,'@E 999,999.99',;
            { |o|nGet1>1000.00},,,,,,T.,,,,,,,,,,'nGet1')

        oButton:=tButton():New(30,10,'fechar',oDlg,{ ||oDlg:End()},100,20,,,,,T.)
    ACTIVATE MSDIALOG oDlg CENTERED
    MsgStop( 'O valor é '+Transform(nGet1,'@E 999,999.00') )
```

Return

## Exemplo utilizando forma equivalente a Classe TGet

```
#include 'protheus.ch'
User Function TesteGet()
Local oDlg, oButton, oCombo, cCombo, nGet1:=0
    DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu Get'
        @10,10 Get oGet1 Var nGet1 Picture "@E 999,999.99" ;
            Size 100,20 Pixel Valid nGet1>1000.0 Of oDlg

        oButton:=tButton():New(30,10,'fechar',oDlg,{ ||oDlg:End()},100,20,,,,,T.)
```

ACTIVATE MSDIALOG oDlg CENTERED

MsgStop( 'O valor é '+Transform(nGet1,'@E 999,999.99') )

Return

## **TButton**

Classe de objetos visuais do tipo botão, o qual permite a utilização.

### **Propriedades:**

IProcessing	Lógico. Se .T. indica o botão está efetuando uma ação.
bAction	Bloco de código. Executado quando o botão é pressionado.

### **Métodos**

**New** - Método construtor da classe.

**Sintaxe** New([anRow], [anCol], [acCaption], [aoWnd], [abAction],[anWidth], [anHeight],;  
[nPar8], [aoFont], [IPar10],[alPixel],[IPar12],[cPar13], [IPar14], [abWhen],  
[bPar16], [IPar17])

Argumento	Obrigat.	Tipo	Descrição
AnRow	Sim	N	Coordenada vertical em pixels ou caracteres.
AnCol	Sim	N	Coordenada horizontal em pixels ou caracteres.
acCaption	Sim	B	Título do botão.
aoWnd	Sim	O	Janela ou controle onde o botão deverá ser criado.
abAction	Não	N	Bloco que deverá ser acionado quando o botão for pressionado.
anWidth	Não	N	Largura do botão em pixels.
anHeight	Não	C	Altura do botão em pixels.
nPar8	Não	B	Reservado.
aoFont	Não	N	Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.
IPar10	Não	N	Reservado.
alPixel	Não	O	Lógico, opcional. Se .T. considera as coordenadas passadas em pixels, se .F. (padrão) considera em caracteres.

lPar12	Não	L	Reservado.
cPar13	Não	O	Reservado.
lPar14	Não	L	Reservado.
abWhen	Não	C	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.
bPar16	Não	L	Reservado.
lPar17	Não	B	Reservado.

## Exemplo

```
#include 'protheus.ch'
User Function TesteGet()
Local oDlg, oButton, oCombo, cCombo, nGet1:=0
    DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu Get'
```

```
    oButton:=tButton():New(30,10,'fechar',oDlg,{||oDlg:End()},100,20,,,,.T.)
```

```
    ACTIVATE MSDIALOG oDlg CENTERED
```

```
    MsgStop( 'O valor é '+Transform(nGet1,'@E 999,999.00') )
```

```
Return
```

## Exemplo utilizando forma equivalente a Classe TButton

```
#include 'protheus.ch'
User Function TesteGet()
Local oDlg, oButton, oCombo, cCombo, nGet1:=0
    DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu Get'
        @10,10 MSGet oGet1 Var nGet1 Picture "@E 999,999.99" ;
            Size 100,20 Pixel Valid nGet1>1000.0 Of oDlg
```

```
        @ 030,010 Button "fechar" Size 100,020 PIXEL OF oDlg action (odlg:end())
```

```
    ACTIVATE MSDIALOG oDlg CENTERED
```

```
    MsgStop( 'O valor é '+Transform(nGet1,'@E 999,999.99') )
```

```
Return
```

## Exercício 00

Fazer uma janela de diálogo como segue:



Obedecer a cores, fontes e formatos

#### Fontes Utilizadas:

Arial 14 Azul Negrito

Verdana 14 Vermelha Negrito

Tahoma 18 Amarela

Script 24 Rosa/Pink

## Objetos de controle

### TCONTROL

Classe abstrata mãe de todos os controles editáveis. tControl é a classe comum entre todos os componentes visuais editáveis.

Propriedades:

IModified	Lógico. Se .T. indica que o conteúdo da variável associada ao controle foi modificado.
IRedOnly	Lógico. Se .T. o conteúdo da variável associada ao controle permanecerá apenas para leitura.
hParent	Numérico. Handle (identificador) do objeto sobre o qual o controle foi criado.
bChange	Bloco de código. Executado quando o estado ou conteúdo do controle é modificado pela ação sobre o controle.



## Métodos

**SETFOCUS** – Força mudança do foco de entrada de dados para o controle.

**Sintaxe:** Setfocus()

## TBUTTON

Utilize a classe TBUTTON para criar um controle visual do tipo botão.

Propriedades:

lProcessing	Lógico. Se .T. indica o botão está efetuando uma ação.
bAction	Bloco de código. Executado quando o botão é pressionado.

## Métodos

**New** - Método construtor da classe.

**Sintaxe:** New ([anRow], [anCol], [acCaption], [aoWnd], [abAction], [anWidth], [anHeight], [nPar8], ;

[AOFONT], [LPAR10], [ALPIXEL],[IPar12],[cPar13], [IPar14], [abWhen], [bPar16], [IPar17])

Argumento	Obrigat.	Tipo	Descrição
anRow	Não	N	Coordenada vertical em pixels ou caracteres.
anCol	Não	N	Coordenada horizontal em pixels ou caracteres.
acCaption	Não	C	Título do botão.
aoWnd	Não	Obj	Janela ou controle onde o botão deverá ser criado.
abAction	Não	Bloco	Bloco que deverá ser acionado quando o botão for pressionado.
anWidth	Não	N	Largura do botão em pixels.
anHeight	Não	N	Altura do botão em pixels.
nPar8			Reservado.
aoFont	Não	Obj	Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.
IPar10			Reservado.

alPixel	Não	Log	Se .T. considera as coordenadas passadas em pixels, se .F. (padrão) considera em caracteres.
lPar12			Reservado.
cPar13			Reservado.
lPar14			Reservado.
abWhen	Não	Bloco	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.
bPar16			Reservado.
lPar17			Reservado.

#### Exemplo

```
#include "protheus.ch"
User Function TesteBut()
Local oDlg, oButton

    DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE "Botoes"
        // Botão para fechar a janela
        oButton:=tButton():New(30,10,"fechar",oDlg,{||oDlg:End()},100,20,,,,
        .T.)
    ACTIVATE MSDIALOG oDlg CENTERED

Return
```

#### Exemplo utilizando forma equivalente a Classe TButton

```
#include "protheus.ch"
User Function TesteBut()
Local oDlg, oButton

    DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE "Botoes"
        @ 30,10, BUTTON PROMPT "fechar" ACTION oDlg:End() SIZE 100,20
        PIXEL of oDlg
    ACTIVATE MSDIALOG oDlg CENTERED

Return
```

## **TCHECKBOX**

Classe de objetos visuais do tipo controle - CheckBox.

**Sintaxe:** New([nRow], [nCol], [cCaption], [bSetGet], [oDlg], [nWidth], [nHeight], [aHelpIds], [bLClicked], [oFont], [bValid], [nClrText], [nClrPane], [lDesign], [lPixel], [cMsg], [lUpdate], [bWhen])

Argumento	Obrigat.	Tipo	Descrição
nRow	Não	N	Coordenada vertical
nCol	Não	N	Coordenada horizontal
cCaption	Sim	C	Texto descritivo
bSetGet	Não	B	Responsável pela setagem de valor
oDlg	Não	O	Nome do objeto da janela a que pertence, ou seja, onde será criado
nWidth	Não	N	Largura do objeto
nHeight	Não	N	Altura do objeto
aHelpIds	Não	A	Array com Help do objeto
bLClicked	Não	B	Executado ao clique do mouse.
oFont	Não	O	Fonte
bValid	Não	B	Executado quando o conteúdo do controle deve ser validado, deve retornar .T. se o conteúdo for válido e .F. quando o conteúdo for inválido.
nClrText	Não	N	Cor do texto da janela
nClrPane	Não	N	Cor de fundo da janela
lDesign	Não	L	Não utilizado
lPixel	Não	L	Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
cMsg	Não	C	Mensagem ao posicionar o mouse sobre o objeto
lUpdate	Não	L	Não utilizado

bWhen	Não	B	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.
-------	-----	---	--

## Exemplo

```
#include 'protheus.ch'
User function BOX01()
```

```
DEFINE MSDIALOG oDlg FROM 0,0 TO 270,400 PIXEL TITLE 'Exemplo da TCBrowse'
    ICheck1 := .T.
```

### //Usando forma nativa da linguagem

```
oCheck1 := TCheckBox():New(01,01,'CheckBox 001',,,oDlg,100,210,,,,,,.T.,,,)
oCheck2 := TCheckBox():New(11,01,'CheckBox 002',,,oDlg,100,210,,,,,,.T.,,,)
oCheck3 := TCheckBox():New(21,01,'CheckBox 003',,,oDlg,100,210,,,,,,.T.,,,)
```

### //Usando forma equivalente

```
@31,01 CheckBox oCheck4 Var IChk2 Prompt 'CheckBox 004' Size 100,210 Pixel
Of oDlg
@41,01 CheckBox oCheck5 Var IChk2 Prompt 'CheckBox 005' Size 100,210
Pixel Of oDlg
```

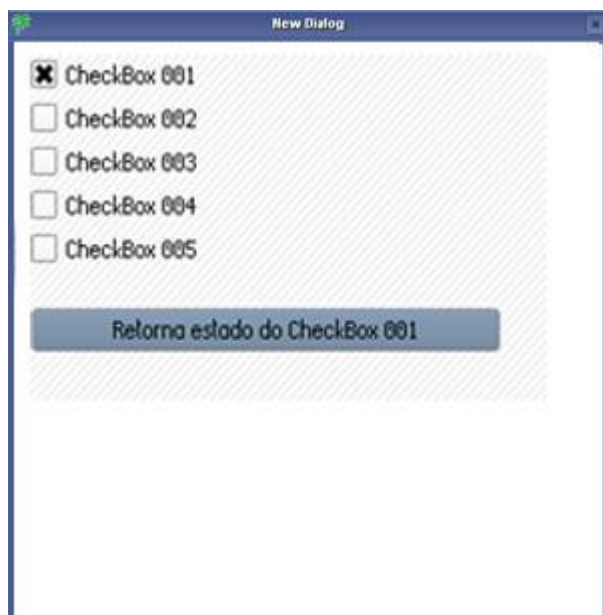
```
// Seta Eventos do primeiro Check
oCheck1:bSetGet := {|| ICheck1 }
oCheck1:bLClicked := {|| ICheck1:=!ICheck1 }
oCheck1:bWhen := {|| .T. }
oCheck1:bValid := {|| Alert('bValid') }
```

### // Principais comandos

```
oBtn := TButton():New( 060, 001, 'Retorna estado do CheckBox 001',,
oDlg,{|| Alert(ICheck1)}, 120, 010,,,F.,.T.,.F.,,.F.,,.F.,,.F. )
ACTIVATE MSDIALOG oDlg CENTERED
```

```
Return
```

## Resultado:



## TCOMBOBOX

Classe de objetos visuais a qual cria uma entrada de dados com múltipla escolha com item definido em uma lista vertical, acionada por F4 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.

Propriedades:

<b>aItems</b>	Array. Lista de itens, caracteres, a serem exibidos. Pode ter os seguintes formatos: a) Seqüencial, exemplo: {"item1","item2",...,"itemN"} ; b) Indexada, exemplo: {"a=item1","b=item2", ..., "n=itemN"}.
<b>nAt</b>	Numérico. Posição do item selecionado.

**Sintaxe:** New([anRow], [anCol], [abSetGet], [anItems], [anWidth], [anHeight], [aoWnd], [nPar8], [abChange], [abValid], [anClrText], [anClrBack], [alPixel], [aoFont], [cPar15], [lPar16], [abWhen], [lPar18], [aPar19], [bPar20], [cPar21], [acReadVar])

Argumento	Obrigat.	Tipo	Descrição
anRow	Não	N	Coordenada vertical em pixels ou caracteres.
anCol	Não	N	Coordenada horizontal em pixels ou caracteres.
abSetGet	Não	B	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. Se a lista for seqüencial, o controle atualizará <var> com o conteúdo do item selecionado, se a lista for indexada, <var> será atualizada com o valor do índice do item selecionado.
anItems	Não	N	Lista de itens, caracteres, a serem exibidos. Pode ter os seguintes formatos: a) Seqüencial, exemplo: {"item1","item2",...,"itemN"} ou b) Indexada, exemplo: {"a=item1","b=item2", ..., "n=itemN"}.
anWidth	Não	N	Largura do controle em pixels.
anHeight	Não	N	Altura do controle em pixels.
aoWnd	Não	O	Janela ou controle onde o controle será criado.
nPar8	Não	N	Reservado.
abChange	Não	B	Executado quando o controle modifica o item selecionado.

abValid	Não	B	Executado quando o conteúdo do controle deve ser validado, deve retornar .T. se o conteúdo for válido e .F. quando o conteúdo for inválido.
anClrBack	Não	N	Cor de fundo do controle.
anClrText	Não	N	Cor do texto do controle.
alPixel	Não	L	Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
aoFont	Não	O	Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do controle.
cPar15	Não	C	Reservado.
lPar16	Não	L	Reservado.
abWhen	Não	B	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	Não	L	Reservado.
aPar19	Não	A	Reservado.
bPar20	Não	B	Reservado.
cPar21	Não	C	Reservado.
acReadVar	Não	C	Nome da variável que o controle deverá manipular, deverá ser a mesma variável informada no parâmetro abSetGet, e será o retorno da função ReadVar( ).

## Exemplo

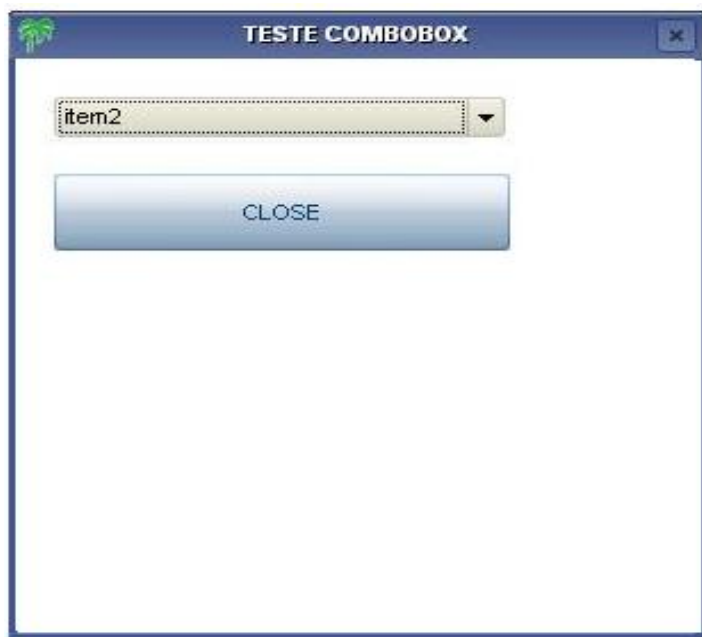
```
#include 'protheus.ch'
User Function CMBBOX1()
Local oDlg, oButton, oCombo, cCombo

aItems:= {'item1','item2','item3'}
cCombo:= aItems[2]
DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'TESTE
COMBOBOX'
oCombo:=
tComboBox():New(10,10,{|u|if(PCount(>0,cCombo:=u,cCombo)},,;
aItems,100,20,oDlg,,{||MsgStop('CHANGE
ITEM')},,,,,,T,,,,,,,,,'cCombo')

// Botão para fechar a janela
```

```
oButton:=tButton():New(30,10,'CLOSE',oDlg,{||oDlg:End()},100,20,,,,.
T.)
ACTIVATE MSDIALOG oDlg CENTERED
MsgStop( 'Item escolhido: '+cCombo )
Return NIL
```

## Resultado:



## Radio

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.

**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

Argumento	Obrigat.	Tipo	Descrição
nLinha,nColuna	Sim	N	Posição do objeto Radio em função da janela em que ele será definido.
oRadio	Sim	O	Objeto do tipo Radio que será criado.
nRadio	Sim	N	Item do objeto Radio que está selecionado.



3D	Não		Item opcional que define se o RADIOBUTTON terá aspecto simples ou 3D.
nLargura,nAltura	Sim	N	Dimensões do objeto Radio.
<ITEMS PROMPT>	Não	A	Utilizar um dos dois identificadores para definir quais os textos que serão vinculados a cada RADIOBUTTON.
cItem1,...,cItemX	Sim	C	Texto que será vinculado a cada RADIOBUTTON.
oObjetoRef	Sim	O	Objeto dialog no qual o componente será definido.
UNIDADE	Não	L	Unidade de medida das dimensões: PIXEL
CHANGE	Não	B	Função ou lista de expressões que será executada na mudança de um item de um RadioButton para outro.
CLICK	Não	B	Função ou lista de expressões que será executada na seleção de um item RADIOBUTTON.

#### Exemplo

```
#include 'protheus.ch'
User Function RADIO1()

Local oDlg, oButton, oRadio, oFld
Local aRadio := {}
Local nRadio := 0, cAtencao := "Escolha feita no radio box"

aAdd( aRadio, "Disco" )
aAdd( aRadio, "Impressora" )
aAdd( aRadio, "Scanner" )

DEFINE MSDIALOG oDlg TITLE "Teste do Radio" FROM 0,0 TO 280,552 OF oDlg PIXEL

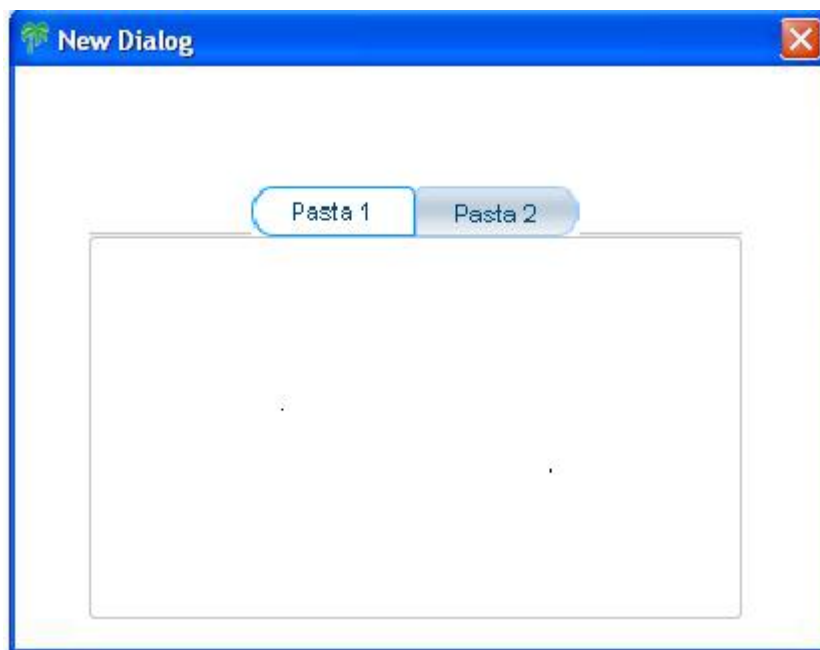
@ 50,06 FOLDER oFld OF oDlg PROMPT "&Pasta1", "&Pasta2" PIXEL SIZE 222,078

@ 30, 10 RADIO oRadio VAR nRadio ITEMS aRadio[1],aRadio[2],aRadio[3] SIZE 65,8 ;
PIXEL OF ;
oFld:aDialogs[1] ;
ON CHANGE ;
(Iif(nRadio==1,MsgInfo("Opção 1",cAtencao),;
Iif(nRadio==2,MsgInfo("Opção 2",cAtencao),MsgInfo("Opção 3",cAtencao))))
```

ACTIVATE MSDIALOG oDIg CENTERED

Return NIL

**Resultado:**



## Editores de tela

Existem diversos editores visuais de tela que geram o código em ADPVL

- GAIA – Desenvolvido pela Microsiga
- MKW - Nativo do Clipper
- Designer - Desenvolvido por Analista de Mercado.

### **Exercicio usando Editor de Tela**

Faça uma tela conforme a seguir utilizando um dos editores de tela disponíveis.



New Dialog

Para prosseguir com a execução /estorno do documento escolha um motivo e detalhe a operação no campo observações .

Usuário: Nome do Usuário

Motivo: 04 - QUANTIDADE INCORRETA

Observações: Corrigida a quantidade do produto XXXXXXXXXXXX  
De: 10  
Para : 20  
Conforme adendo número 9999999da proposta XXXXXXXXXXXX

Confirma exclusão de

Cancelar Ok

### Premissas:

Usuário deve trazer o nome do usuário logado.

Criar a tabela de motivos no SX5 – Tabela Z9.

Criar uma arquivo com os seguintes campos: SZ0

Filial C 2 / Motivo C 2 / Observação M 10 / Usuario C 15

Na combo do motivo deve trazer os itens da tabela Z9.

O botão “sim” deve ser habilitado somente se a observação estiver preenchida e o motivo for diferente de nenhum,

Quando o botão “sim” for clicado as informações da tela deverão ser gravadas na tabela SZ0.

**Montando Janelas****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).

**LISTBOX SIMPLES****Sintaxe:**

```
@ nLin,nCol LISTBOX oListBox VAR nLista ITEMS aLista SIZE nLarg,nAlt OF oDlg PIXEL ON  
CHANGE CHANGE
```

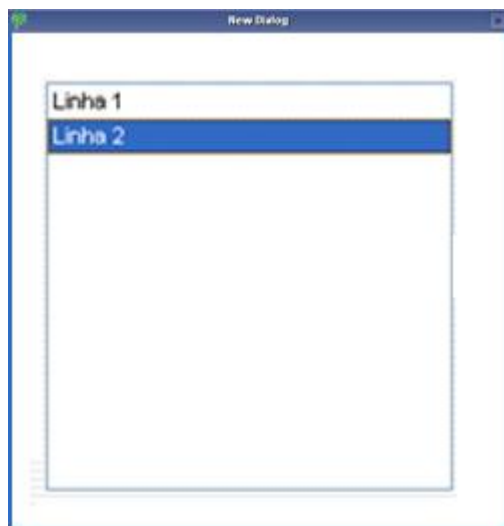
**Atributos:**

<b>nLin,nCol</b>	Posição do objeto ListBox em função da janela em que ele será definido.
<b>oListBox</b>	Objeto ListBox que será criado.
<b>nLista</b>	Variável numérica que guarda o item selecionado no ListBox.
<b>aLista</b>	Array simples com os itens da lista.
<b>nLarg,nAlt</b>	Dimensões do objeto ListBox.
<b>oObjetoRef</b>	Objeto dialog no qual o componente será definido.
<b>PIXEL</b>	Se informado, as dimensões serão em pixel.
<b>CHANGE</b>	Lista de expressões que será executada na seleção de um item do ListBox.

**Exemplo**

```
#INCLUDE "PROTHEUS.CH"  
User Function listbox1()  
Local OLISTBOX  
Private nLista := 2
```

```
Private aLista := {"Linha 1", "Linha 2"}  
DEFINE MSDIALOG _oDlg TITLE "ListBox" FROM (210),(210) TO (468),(520)  
PIXEL  
    @ 05,05 LISTBOX oList VAR nLista ITEMS aLista PIXEL SIZE 100,100 OF _oDlg  
;  
    ON CHANGE MSGINFO(nLista)  
    ACTIVATE MSDIALOG _oDlg CENTERED  
Return
```

**Resultado:**

Ao Clicar no Item da lista aparece a mensagem seguinte.



## LISTBOX MULTIPLOS

### Sintaxe:

@nLin,nCol LISTBOX oListBox VAR nLista FIELDS HEADER "Header1" ... "HeaderX";  
SIZE nLarg,nAlt OF oDlg PIXEL ON CHANGE CHANGE

### Atributos:

<b>nLin,nCol</b>	Posição do objeto ListBox em função da janela em que ele será definido.
<b>oListBox</b>	Objeto ListBox que será criado.
<b>nLista</b>	Variável numérica que guarda o item selecionado no ListBox.
<b>"Header1",...,"HeaderX"</b>	Strings identificando os títulos das colunas do Grid.
<b>nLarg,nAlt</b>	Dimensões do objeto ListBox.
<b>oObjetoRef</b>	Objeto dialog no qual o componente será definido.
<b>PIXEL</b>	Se informado, as dimensões serão em pixel.
<b>CHANGE</b>	Lista de expressões que será executada na seleção de um item do ListBox.
<b>bLine</b>	Atributo do objeto ListBox que vincula cada linha,coluna do array, com cada cabeçalho do grid.

### Métodos:

<b>SetArray()</b>	Método o objeto ListBox que define qual o array contém os dados que serão exibidos no grid.
-------------------	---

### Exemplo

```
User Function listbox2() // Figura 1
Local OLIST
Private nLista := 0
Private aLista := {"Linha 1 a", "Linha 1 b"}, {"Linha2 a", "Linha 2 b"}
DEFINE MSDIALOG _oDlg TITLE "ListBox" FROM (210),(210) TO (330),(420)
PIXEL
    // Cria Componentes Padroes do Sistema
    @ 05,05 LISTBOX oList FIELDS HEADER "Coluna 1" ,"Coluna 2" PIXEL SIZE
    100,50 OF _oDlg
    oList:SetArray( aLista )
    oList:bLine := {|| { aLista[oList:nAt,1],,
                        aLista[oList:nAt,2]}}
    ACTIVATE MSDIALOG _oDlg CENTERED
```

Return

### Exemplo

User Function listbox3()// **Figura 2**

Local OLIST

Local oOk := LoadBitmap( GetResources(), "LBOK" )

Local oNo := LoadBitmap( GetResources(), "LBNO" )

Private nLista := 0

Private aLista := {{.f., "Linha 1 a", "Linha 1 b"}, {f., "Linha2 a", "Linha 2 b"}}

DEFINE MSDIALOG \_oDlg TITLE "ListBox" FROM (210),(210) TO (390),(480)

PIXEL

// Cria Componentes Padroes do Sistema

@ 05,05 LISTBOX oList FIELDS HEADER "", "Coluna 1", "Coluna 2" PIXEL SIZE 100,80 OF \_oDlg;

ON dblClick(aLista[oList:nAt,1] := !aLista[oList:nAt,1],oList:Refresh())

oList:SetArray( aLista )

oList:bLine := {|| {Iif(aLista[oList:nAt,1],oOk,oNo),;

aLista[oList:nAt,2],;

aLista[oList:nAt,3]}}

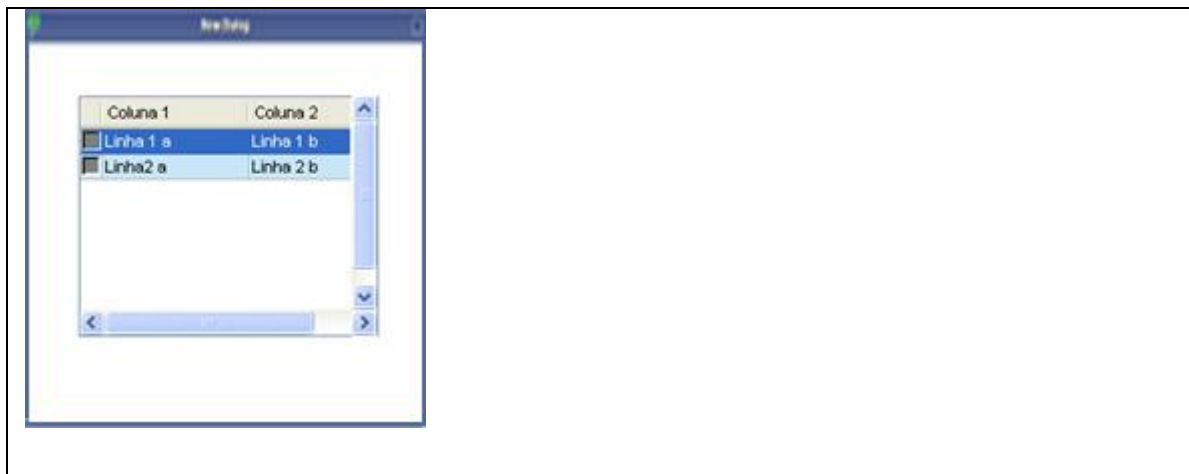
ACTIVATE MSDIALOG \_oDlg CENTERED

Return

### Figura 1



### Figura 2





## Exercício ListBox

1. Fazer um programa que monte uma combo de todas as tabelas (x5\_tabela = '00')

Na escolha de cada tabela deve montar um listbox com o todas as chaves e a descrição da tabela escolhida, o listbox deve ser dinâmico.

2. Fazer uma tela muito parecida com o listbox do exercício anterior.

Criar combo com código e nome do cliente

Montar a listbox de acordo com o cliente escolhido pelo usuário na combo, contendo todos os títulos do cliente.

Prefixo

Numero

Parcela

Tipo

Emissão

Vencido

Valor

Saldo

## **DBTREE**

Classe Advpl que permite a criação de um objeto visual do tipo Tree.

### **Métodos principais:**

<b>New</b>	Contrutor da Classe Dbtree. Retorna uma nova instância do Objeto da Classe Dbtree.
<b>AddTree</b>	Através do método AddTree, é possível acrescentar um 'nó' na árvore atual. Uma vez acrescentado o nó, as próximas inclusões de itens na árvore serão realizadas neste nó, até que o mesmo seja fechado através do método Endtree.
<b>AddTreeItem</b>	Através dele, podemos acrescentar itens na árvore atual ou último nó acrescentado em aberto.
<b>EndTree</b>	Deve ser utilizado após todos os elementos e itens serem acrescentados a um nó da árvore, aberto com o método AddTree().

**Sintaxe:** DbTree():New ( [ nTop ] , [ nLeft ] , [ nBottom ] , [ nRight ] , [ oWnd ] , [ bchange ]  
,  
[ bRClick ] , [ ICargo ] , [ IDisable ] )

Argumento	Obrigat.	Tipo	Descrição
nTop	Sim	N	Coordenada vertical superior do Objeto.
nLeft	Não	N	Coordenada horizontal esquerda do Objeto.
nBottom	Não	N	Coordenada vertical inferior do Objeto.
nRight	Não	N	Coordenada horizontal direita do Objeto.
oWnd	Não	O	Janela pai do Objeto Tree
bchange	Não	B	Code-Block contendo a ação a ser executada na mudança de foco entre os elementos da árvore.
bRClick	Não	B	Code-Block a ser executado quando pressionado o botão direito do Mouse sobre um elemento da árvore.
ICargo	Não	L	Se .T., indica que os elementos do Tree utilizarão a propriedade CARGO, capaz de armazenar uma string identificadora, fornecida na montagem para cada elemento e item da árvore.
IDisable	Não	L	Se .T., cria o objeto do Tree desabilitado, não permitindo foco e navegação no mesmo qneuando ele não seja

			habilitado.
--	--	--	-------------

## Enchoice

Classe Advpl que permite a criação de um objeto visual de tela de captura de informações compostas por múltiplos campos digitáveis acompanhados de seus respectivos textos explicativos.

Sintaxe:

```
Enchoice( cAlias, nReg, nOpc, aCRA, cLetra, cTexto, aAcho, aPos, aCpos, nModelo, nColMens,
;
      cMensagem, cTudoOk, oWnd, IF3, IMemoria, IColumn, caTela, INoFolder, IProperty)
```

Argumento	Obrigat.	Tipo	Descrição
cAlias	Sim	C	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
nReg	Sim	N	Parâmetro não utilizado
nOpc	Sim	N	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização)
aCRA	Não	A	Parâmetro não utilizado
cLetra	Não	C	Parâmetro não utilizado
cTexto	Não	C	Parâmetro não utilizado
aAcho	Não	A	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	Não	A	Vetor com coordenadas para criação da enchoice no formato {<top>, <left>, <bottom>, <right>}
aCpos	Não	A	Vetor com nome dos campos que poderão ser editados
nModelo	Não	N	Se for diferente de 1 desabilita execução de gatilhos estrangeiros
nColMens	Não	N	Parâmetro não utilizado
cMensagem	Não	C	Parâmetro não utilizado
cTudoOk	Não	C	Expressão para validação da Enchoice
oWnd	Sim	O	Objeto (janela, painel, etc.) onde a enchoice será criada.
IF3	Não	L	Indica se a enchoice esta sendo criada em uma consulta F3 para utilizar variáveis de memória

IMemoria	Não	L	Indica se a enchoice utilizará variáveis de memória ou os campos da tabela na edição
IColumn	Não	L	Indica se a apresentação dos campos será em forma de coluna
caTela	Não	C	Nome da variável tipo "private" que a enchoice utilizará no lugar da propriedade aTela
INoFolder	Não	L	Indica se a enchoice não irá utilizar as Pastas de Cadastro (SXA)
IProperty	Não	L	Indica se a enchoice não utilizará as variáveis aTela e aGets, somente suas propriedades com os mesmos nomes

#### Exemplo

```
User Function TENCHO01()
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 IF3       := .F.
```

**Continuação...**

```
Local IMemoria      := .T.
Local IColumn       := .F.
Local caTela        := ""
Local INoFolder     := .F.
Local IProperty     := .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, , , , aCpoEnch, aPos, aAlterEnch, nModelo, , ;
        ,, oDlg, IF3, IMemoria, IColumn, caTela, INoFolder, IProperty)

ACTIVATE MSDIALOG oDlg CENTERED

Return
```

## Resultado:

**Detalhes**

Codigo

Loja

Nome

N Fantasia

Tipo

DDI

Regiao

Cod. Abics

01

01

PRIMEIRO CLIENTE

CLIENTE

CONS.FINAL

Dt Ini Vincu

Dt Fim Vincu

Usa DDA

Opt. Simples

Cod.Mun.SIAF

Contr. TARE ?

Recolhe IRRF

TDA

//

//

NAO

**Cadastro de Clientes**

[Localizar](#)
[Filtrar](#)
[Imprimir](#)
[Configurar](#)

Codigo + Loja

Buscar

Codigo	Loja	Nome	N Fantasia	Tipo	DDI	Regiao	C
01	01	PRIMEIRO CLIENTE	CLIENTE	CONS.FINAL			

[Visualizar](#)
[Sair](#)
[Ações relacionadas](#)

## MSGetDados

Classe de objetos visuais que 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)
```

Argumento	Obrigat.	Tipo	Descrição
nTop	Sim	N	Distancia entre a MsGetDados e o extremidade superior do objeto que a contém.
nLeft	Sim	N	Distancia entre a MsGetDados e o extremidade esquerda do objeto que a contém.
nBottom	Sim	N	Distancia entre a MsGetDados e o extremidade inferior do objeto que a contém.
nRight	Sim	N	Distancia entre a MsGetDados e o extremidade direita do objeto que a contém.
nOpc	Sim	N	Posição do elemento do vetor aRotina que a MsGetDados usará como referencia.

cLinhaOk	Não	C	Função executada para validar o contexto da linha atual do aCols.
cTudoOk	Não	C	Função executada para validar o contexto geral da MsGetDados (todo aCols).
cIniCpos	Não	C	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	Não	L	Habilita excluir linhas do aCols. Valor padrão falso.
aAlter	Não	A	Vetor com os campos que poderão ser alterados.
uPar1	Não		Parâmetro reservado.
lEmpty	Não	L	Habilita validação da primeira coluna do aCols para esta não poder estar vazia. Valor padrão falso.
nMax	Não	N	Número máximo de linhas permitidas. Valor padrão 99.
cFieldOk	Não	C	Função executada na validação do campo.
cSuperDel	Não	C	Função executada quando pressionada as teclas <Ctrl>+<Delete>.
uPar2	Não		Parâmetro reservado.
cDelOk	Não	C	Função executada para validar a exclusão de uma linha do aCols.
oWnd	Não	O	Objeto no qual a MsGetDados será criada.

Variáveis private:

<b>aRotina</b>	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: Pesquisar, Visualizar, Incluir, Alterar, Excluir
<b>aHeader</b>	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.
<b>lRefresh</b>	Variável tipo lógica para uso reservado.

Variáveis públicas:

<b>N</b>	Indica qual a linha posicionada do aCols.
----------	---

Funções de validação:

<b>cLinhaOk</b>	Função de validação na mudança das linhas da grid. Não pode ser definida como Static Function.
<b>cTudoOk</b>	Função de validação da confirmação da operação com o grid. Não pode ser definida como Static Function.

Métodos adicionais:

<b>ForceRefresh()</b>	Atualiza a MsGetDados com a tabela e posiciona na primeira linha.
<b>Hide()</b>	Ocultar a MsGetDados.
<b>Show()</b>	Mostra a MsGetDados.

Exemplo

```
User Function GTDA01()

Local nI
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", "AxAltera", 0, 4},,
    {"Excluir", "AxDeleta", 0, 5}}
SX3->(DbSetOrder(1))
SX3->(DbSeek("SA1"))
```

**Continua...**



**Continuação...**

```
While SX3->(!Eof()) .and. SX3->X3_ARQUIVO == "SA1"
  If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL
    nUsado++
    AAdd (aHeader, {Trim(SX3->X3_TITULO),; //01 - titulo
                  SX3->X3_CAMPO      ,; //02- nome do campo
                  SX3->X3_PICTURE    ,; //03 - mascara do campo
                  SX3->X3_TAMANHO    ,; //04 - tamanho
                  SX3->X3_DECIMAL     ,; //05 - decimais
                  SX3->X3_VALID       ,; //06 - validacao
                  SX3->X3_USADO       ,; //07 - USADO
                  SX3->X3_TIPO        ,; //08 - TIPO DO CAMPO
                  SX3->X3_ARQUIVO     ,; //09 - ALIAS
                  SX3->X3_CONTEXT})   //10 - Virtual ou Real
    EndIf
    SX3->(DbSkip())
  End

AADD(aCols,Array(nUsado+1))

For nI := 1 To nUsado
  aCols[1][nI] := CriaVar(aHeader[nI][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,,,,.T.,{"A1_NOME"},,.F.,200,,,,oDlg)

oDlg:ICentered := .T.
oDlg:Activate()

Return
```

**MSNEWGETDADOS**

Classe de objetos visuais que permite a criação de um grid digitável com uma ou mais colunas, baseado em um array com a diferença que esta classe permite várias instâncias de GetDados pois não requer aHeader e aCols do tipo private.

#### Sintaxe:

MSNEWGETDADOS():New(nSuperior, nEsquerda, nInferior, nDireita, nOpc, cLinOk, cTudoOk, cIniCpos, aAlterGDa, nFreeze, nMax, cFieldOk, cSuperDel, cDelOk, oDLG, aHeader, aCols)

Argumento	Obrigat.	Tipo	Descrição
nSuperior	Sim	N	Distancia entre a MsNewGetDados e o extremidade superior do objeto que a contem
nEsquerda	Sim	N	Distancia entre a MsNewGetDados e o extremidade esquerda do objeto que a contem
nInferior	Sim	N	Distancia entre a MsNewGetDados e o extremidade inferior do objeto que a contem
nDireita	Sim	N	Distancia entre a MsNewGetDados e o extremidade direita do objeto que a contem
nOpc	Sim	N	Operação em execução: 2- Visualizar, 3- Incluir, 4- Alterar, 5- Excluir
cLinOk	Não	C	Função executada para validar o contexto da linha atual do aCols
cTudoOk	Não	C	Função executada para validar o contexto geral da MsNewGetDados (todo aCols)
cIniCpos	Não	C	Nome dos campos do tipo caracter que utilizarão incremento automático.
aAlterGDa	Não	A	Campos alteráveis da GetDados
nFreeze	Não	N	Campos estáticos na GetDados, partindo sempre da posição inicial da getdados aonde: 1- Primeiro campo congelado 2- Primeiro e segundo campos congelados...
nMax	Não	N	Número máximo de linhas permitidas. Valor padrão 99
cFieldOk	Não	C	Função executada na validação do campo
cSuperDel	Não	C	Função executada quando pressionada as teclas <Ctrl> + <Delete>
cDelOk	Não	C	Função executada para validar a exclusão de uma linha do aCols
oDLG	Sim	O	Objeto no qual a MsNewGetDados será criada
aHeader	Sim	A	Array a ser tratado internamente na MsNewGetDados como aHeader

aCols	Sim	C	Array a ser tratado internamente na MsNewGetDados como aCols
-------	-----	---	--

#### Exemplo

```
#include 'protheus.ch'
User Function NEWGTDA1()
Private cCadastro := "Contatos"
Private aRotina := { {"Pesquisar","AxPesqui",0,1} ,;
                    {"Visualizar","u_CTGERAL",0,2} }
    mBrowse( 6,1,22,75,"SA1")

Return

User Function CTGERAL(cAlias, nReg, nOpc)

Private cOpc

SetPrvt("oDlg","oFld1","oGetHor","oBrw2")

// Variaveis que definem a Acao do Formulario
Private VISUAL := .F.
Private INCLUI := .F.
Private ALTERA := .F.
Private DELETA := .F.
Private oFolder1
Private aFolder1 := {"CLIENTES"}

// Privates das NewGetDados

Private noGetZ7 := 0

Private aHeaderZ7 := {}
Private acolsZ7 := {}
Private aColsCP := {}

aButs := {}

aHeaderZ7()
acolsZ7(nReg,nOpc)

DEFINE MSDIALOG oDlg TITLE "Teste do Msnewgetdados" FROM 150,150 TO 565,925
```

PIXEL

```
oGetHor := MsNewGetDados():New(13,03,193,388,cOpc,"AllwaysTrue",;  
    "AllwaysTrue",,,,000,999,,,,oDlg,aHeaderZ7,acolsZ7)
```

ACTIVATE MSDIALOG oDlg CENTERED

Return(.T.)

Static Function aHeaderZ7()

DbSelectArea("SX3")

DbSetOrder(1)

DbSeek("SA1")

**Continua...**

**Continuação...**

```
While !Eof() .and. SX3->X3_ARQUIVO == "SA1"
  If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL
    noGetZ7++
    Aadd(aHeaderZ7,{ Trim(SX3->X3_TITULO),,; //01 - titulo
                    SX3->X3_CAMPO      ,; //02- nome do campo
                    SX3->X3_PICTURE    ,; //03 - mascara do campo
                    SX3->X3_TAMANHO    ,; //04 - tamanho
                    SX3->X3_DECIMAL    ,; //05 - decimais
                    SX3->X3_VALID      ,; //06 - validacao
                    SX3->X3_USADO      ,; //07 - USADO
                    SX3->X3_TIPO       ,; //08 - TIPO DO CAMPO
                    SX3->X3_ARQUIVO    ,; //09 - ALIAS
                    SX3->X3_CONTEXT})   //10 - Virtual ou Real
  EndIf
  SX3->(DbSkip())
Enddo
SX3->(dbclosearea())
```

Return

Static Function acolsZ7(nReg,nOpc)

```
acolsZ7:={}
```

```
dbselectarea("SA1")
```

```
SA1->(dbSetOrder(1))
```

```
IAchei := SA1->(dbSeek(xFilial("SA1")))
```

```
AADD(acolsZ7,Array(Len(aHeaderZ7)+1))
```

```
For nI:= 1 To Len(aHeaderZ7)
```

```
  If aHeaderZ7[nI][10] <> 'V'
```

```
    acolsZ7[LEN(acolsZ7)][nI] := SA1->&(aHeaderZ7[nI][2])
```

```
  Else
```

```
    acolsZ7[LEN(acolsZ7)][nI] := Criavar(aHeaderZ7[nI][2])
```

```
  Endif
```

```
Next
```

```
acolsZ7[LEN(acolsZ7)][Len(aHeaderZ7)+1] := .F. // Indica se a linha esta deletada
```

Return

## **MSSELECT**

Classe que cria um objeto browse (ou grid), com a primeira coluna sendo do tipo marcação.

### **Propriedades:**

oWnd	Objeto pai da MsSelect (MSDIALOG, MSWINDOW, MSPANEL)
oBrowse	Objeto browse (ou grid) da MsSelect, herdado da classe TCBROWSE

### **Sintaxe:**

```
MsSelect():NEW ( < cAlias > , [ cCampo ] , [ cCpo ] , [ aCampos ] , [ lInv ] , [ cMar ] , [ aCord ] , ;  
[ cTopFun ] , [ cBotFun ] , [ oWnd ] , [ reserved ] , [ aColors ] )
```

Argumento	Obrigat.	Tipo	Descrição
cAlias	Sim	C	Tabela que será utilizada pela MsSelect
cCampo	Sim	C	Campo que será utilizado na coluna de marcação
cCpo	Sim	C	Campo da tabela informada pelo parametro cAlias ou função que será executada na apresentação da coluna que indica se a linha da MsSelect esta habilitada ou não
aCampos	Sim	A	Vetor com informações dos campos para criação das colunas da MsSelect. Se não for informado, a MsSelect irá criar as colunas baseado no Dicionário de Campos (SX3) da tabela informada no parametro cAlias. Formato: 1 - campo ou bloco de código com conteúdo que será apresentado na coluna pela MsSelect 2 - não utilizado 3 - cabeçalho 4 - máscara de apresentação (picture)
lInv	Não	L	Indica se MsSelect irá utilizar marcação invertida
cMar	Sim	C	Marca que será utilizada pela MsSelect para controle do campo informado pelo parametro cCampo. Para utilizar o parametro cMar, utilize a função GetMark para retornar a próxima marca.
aCord	Não	A	Coordenadas para criação da MsSelect. Formato: 1 - Linha inicial 2 - Coluna inicial 3 - Linha final 4 - Coluna final

cTopFun	Não	C	Função que retornará o conteúdo inicial que a MsSelect utilizará para apresentar a primeira linha da tabela, como um range, junto com o parametro cBotFun. O conteúdo retornado será utilizado para fazer o posicionamento da tabela informada pelo parametro cAlias, baseado na chave de índice posicionada para a mesma.
cBotFun	Não	C	Função que retornará o conteúdo final que a MsSelect utilizará para apresentar a última linha da tabela, como um range, junto com o parametro cTopFun. O conteúdo retornado será utilizado para o posicionamento final da tabela informada pelo parametro cAlias, baseado na chave de índice posicionada para a mesma
oWnd	Não	O	Objeto pai da MsSelect (MSDIALOG, MSWINDOW, MSPANEL, etc.)
reserved	Não		Parametro reservado
aColors	Não	A	Vetor com regras para a apresentação da coluna de legenda. Formato: 1 - Expressão ADVPL com retorno lógico que indica se a cor será utilizada pela coluna 2 - cor da coluna: BR_AMARELO BR_AZUL BR_BRANCO BR_CINZA BR_LARANJA BR_MARROM BR_PINK BR_PRETO BR_VERDE BR_VERMELHO

#### Exemplo

```
#include 'protheus.ch'
User Function MSELECT1()
Local oDlg, oMark
Local cMarca:= GetMark(), lInverte:= .T.

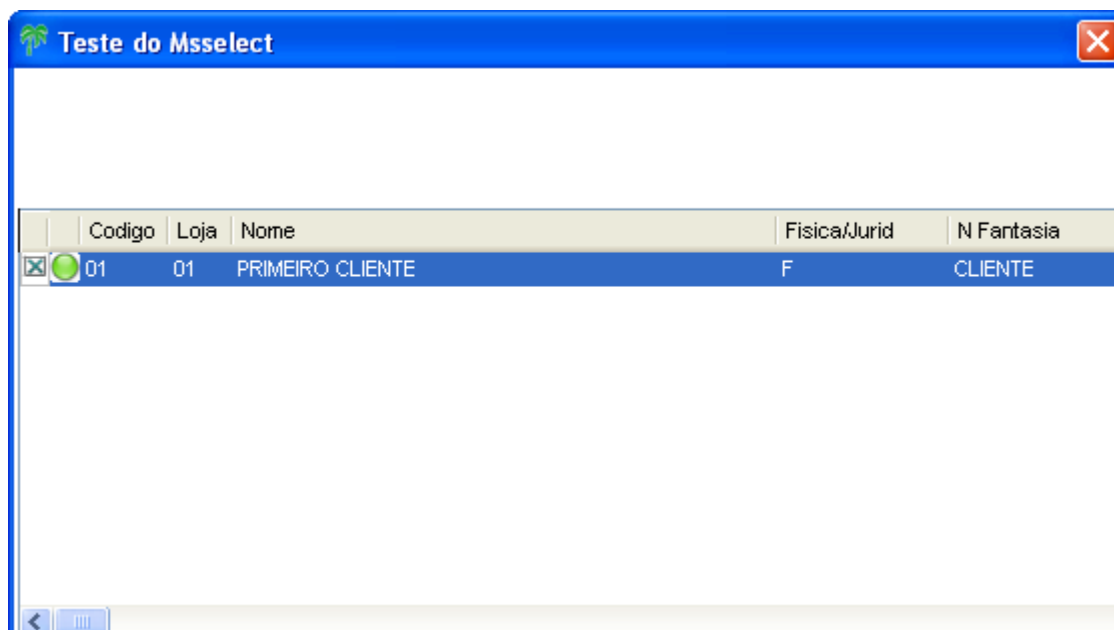
DEFINE MSDIALOG oDlg TITLE "Teste do Msselect" FROM 0,0 TO 280,552 OF oDlg
PIXEL

oMark :=
MsSelect():New("SA1","A1_OK","!A1_COD",,@lInverte,@cMarca,{35,1,143,315})


ACTIVATE MSDIALOG oDlg CENTERED
```

Return NIL

**Resultado:**



Teste do Msselect

	Codigo	Loja	Nome	Fisica/Jurid	N Fantasia
	01	01	PRIMEIRO CLIENTE	F	CLIENTE



## **MSGETDB**

Classe de objetos visuais que 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)

Argumento	Obrigat.	Tipo	Descrição
nTop	Sim	N	Distancia entre a MSGETDB e o extremidade superior do objeto que a contém.
nLeft	Sim	N	Distancia entre a MSGETDB e o extremidade esquerda do objeto que a contém.
nBottom	Sim	N	Distancia entre a MsGetDB e o extremidade inferior do objeto que a contém.
nRight	Sim	N	Distancia entre a MsGetDB e o extremidade direita do objeto que a contém.
nOpc	Sim	N	Posição do elemento do vetor aRotina que a MsGetDB usará como referência.
cLinhaOk	Sim	C	Função executada para validar o contexto da linha atual do aCols.
cTudoOk	Sim	C	Função executada para validar o contexto geral da MsGetDB (todo aCols).
cIniCpos	Sim	C	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	Sim	L	Habilita a opção de excluir linhas do aCols. Valor padrão falso.
aAlter	Sim	A	Vetor com os campos que poderão ser alterados.
nFreeze	Sim	N	Indica qual coluna não ficara congelada na exibição.
lEmpty	Sim	L	Habilita validação da primeira coluna do aCols para esta não poder estar vazia. Valor padrão falso.
uPar1	Sim		Parâmetro reservado.
cFieldOk	Sim	C	Função executada na validação do campo.
cTRB	Sim	C	Alias da tabela temporária.
lCondicional	Sim	L	Reservado

lAppend	Sim	L	Indica se a MsGetDB ira criar uma linha em branco automaticamente quando for inclusão.
cDelOk	Sim	C	Função executada para validar a exclusão de uma linha do aCols.
lDisparos	Sim	L	Indica se será utilizado o Dicionário de Dados para consulta padrão, inicialização padrão e gatilhos.
uPar2	Sim		Parâmetro reservado.
cSuperDel	Sim	C	Função executada quando pressionada as teclas <Ctrl> + <Delete>.
oWnd	Sim	O	Objeto no qual a MsGetDB será criada.

Variáveis private:

<b>aRotina</b>	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: Pesquisar, Visualizar, Incluir, Alterar, Excluir
<b>aHeader</b>	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.
<b>lRefresh</b>	Variável tipo lógica para uso reservado.

Variáveis públicas:

<b>nBrLin</b>	Indica qual a linha posicionada do aCols.
---------------	---

Funções de validação:

<b>cLinhaOk</b>	Função de validação na mudança das linhas da grid. Não pode ser definida como Static Function.
<b>cTudoOk</b>	Função de validação da confirmação da operação com o grid. Não pode ser definida como Static Function.

Métodos adicionais:

<b>ForceRefresh()</b>	Atualiza a MsGetDados com a tabela e posiciona na primeira linha.
-----------------------	---

Exemplo

```
#include "protheus.ch"
User Function GETDB01()
```

```

Local Ni, 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}}

```

## Continuação...

```

SX3->(DbSetOrder(1))
SX3->(DbSeek("SA1"))
While SX3->(!Eof()) .and. SX3->X3_ARQUIVO == "SA1"
    If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL
        noGetZ7++
        Add(aHeaderZ7,{ Trim(SX3->X3_TITULO),, //01 - titulo
                        SX3->X3_CAMPO      ,; //02- nome do campo
                        SX3->X3_PICTURE    ,; //03 - mascara do campo
                        SX3->X3_TAMANHO    ,; //04 - tamanho
                        SX3->X3_DECIMAL     ,; //05 - decimais
                        SX3->X3_VALID      ,; //06 - validacao
                        SX3->X3_USADO       ,; //07 - USADO
                        SX3->X3_TIPO        ,; //08 - TIPO DO CAMPO
                        SX3->X3_ARQUIVO     ,; //09 - ALIAS
                        SX3->X3_CONTEXT})   //10 - Virtual ou Real
    EndIf
    SX3->(DbSkip())
Enddo
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,,, "+A1_COD",,
.T.,{"A1_NOME"},1,.F.,cCriaTrab,,,T.,oDlg,.T.,,,)

```

```
oDlg:ICentered := .T.  
oDlg:Activate()  
DbSelectArea(cCriaTrab)  
DbCloseArea()
```

Return

## Resultado:



## Relatórios Gráficos

### TMSPRINTER

Cria um novo relatório.

#### Métodos:

<b>New</b>	Método construtor da classe.
<b>SetPortrait</b>	Define orientação de página do relatório como retrato
<b>SetLandscape</b>	Define a orientação de página do relatório como paisagem
<b>StartPage</b>	Inicializa uma nova página de impressão
<b>EndPage</b>	Finaliza um página de impressão
<b>End</b>	Encerra o objeto
<b>Say</b>	Imprime texto nas coordenadas informadas pelos parametros, utilizando as especificações do objeto TFont, cor da fonte e cor de fundo
<b>SayBitmap</b>	Imprime uma imagem nas coordenadas informadas pelos parametros
<b>Line</b>	Desenha uma linha baseado nas coordenadas informadas pelos parametros, utilizando as especificações do objeto TPen.
<b>Box</b>	Desenha uma caixa baseado nas coordenadas informadas pelos parametros, utilizando as especificações do objeto TPen.
<b>FillRect</b>	Preenche um retangulo na impressão utilizando as especificações do objeto TBrush.
<b>Preview</b>	Apresenta preview do relatório.
<b>Setup</b>	Mostra tela de Configuração de impressao
<b>Print</b>	Imprime o que já estiver instanciado
<b>ThinLine</b>	Desenha uma linha simples iniciando na linha atual de impressão com a largura da página
<b>FatLine</b>	Desenha uma linha com altura maior que a do método ThinLine, iniciando na linha atual de impressão com a largura da página

Exemplo

```
# INCLUDE "PROTHEUS.CH"
```

USER FUNCTION relgra()

PRIVATE cData := dtoc(Date())

PRIVATE cTime := time()

PRIVATE cTitulo := "Impressão - Relatório de produtos"

PRIVATE oPrn, oArial07, oArial09B ,oArial14B

**Continua...**

**Continuação...**

Private nPag := 0

```
DEFINE FONT oAria10 NAME "Arial" SIZE 0,10 OF oPrn
DEFINE FONT oAria10B NAME "Arial" SIZE 0,10 OF oPrn BOLD
DEFINE FONT oAria12B NAME "Arial" SIZE 0,12 OF oPrn BOLD
DEFINE FONT oAria20B NAME "Arial" SIZE 0,16 OF oPrn BOLD
```

```
oPrn := TMSPrinter():New(cTitulo)
oPrn:SetPortrait()
```

```
meucabec()
itens()
oPrn:EndPage()
oPrn:End()
oPrn:Preview()
oPrn:End()
```

Return

STATIC FUNCTION itens()

```
Local nCont := 0
Local nLin := 300
Local cTipo := " "
```

```
BeginSQL Alias "TRB"
  %NoParser%
  Select B1_COD, B1_DESC, B1_UM, B1_TIPO from %Table:SB1% B1
  ORDER BY B1_TIPO,B1_UM
EndSQL
```

TRB->(dbGoTop())//COLOCA-SE O APELIDO ANTES DE QQ COMANDO DE BANCO DE DADOS POR PRECAUCAO

```
ctipo := TRB->B1_UM
While TRB->(!EOF())// TRB->(<COMANDO>)
```

```
  If nLin > 2000
    oPrn:EndPage()
    meucabec()
    nLin := 300
  oPrn:Say(nLin,0020,TRB->B1_COD,oAria10)
  oPrn:Say(nLin,0250,TRB->B1_DESC,oAria10)
  oPrn:Say(nLin,1040,TRB->B1_UM,oAria10)
Endif
```

**Continua...**

**Continuação...**

```
oPrn:Say(nLin,1240,TRB->B1_TIPO,oArial10)
nCont := nCont + 1
nLin := nLin + 50

TRB->(dbSkip()) // Avanca o ponteiro do registro no arquivo
if cTipo <> TRB->B1_UM
    nLin := nLin + 50
    oPrn:Say(nLin,0020,"Total por Unidade : "+cTipo + " - ";
        +cValtochar(nCont),oArial10B,,CLR_RED)
    nCont := 0
    cTipo := TRB->B1_UM
    nLin := nLin + 150
Endif
Enddo

TRB->(dbCloseArea())

Return

Static function meucabec()
    nPag++
    oPrn:StartPage()

    oPrn:SayBitmap(nLinIni1,nColIni1,"FORD.bmp",nLinFim1,nColFim1)
    oPrn:Say(0020,0600,"RELATÓRIO DO CADASTRO DE PRODUTO",oArial20B)
    oPrn:Say(0100,1040,"DATA:",oArial12B)
    oPrn:Say(0100,1240,cData,oArial12B)
    oPrn:Say(0100,1500,"HORA:",oArial12B)
    oPrn:Say(0100,1700,cTime,oArial12B)
    oPrn:Say(0100,1900,"PAGINA:",oArial12B)
    oPrn:Say(0100,2100,cvaltochar(nPag),oArial12B)

    oPrn:Line(0150,0000,0150,3000)
    oPrn:Say(0200,0020,"Cod.Produto",oArial10b)
    oPrn:Say(0200,0250,"Produto",oArial10b)
    oPrn:Say(0200,1040,"Unidade",oArial10b)
    oPrn:Say(0200,1240,"Tipo",oArial10b)
    oPrn:Line(0250,0000,0250,3000)
Return
```



## **Exercício TMSPrinter**

Incluir um relatório de Clientes, relacionando todo o seu faturamento e seus títulos a receber, a cada quebra deverá ter uma separação com uma linha (usem o método line() )

Logo 4 esquerda

Data

Hora

Pagina

Cabeçalho em negrito e separado por linhas(line())

Quebrar por Cliente

totalizar, Valor Total da Nota, Valor do ICMS, Valor do IPI

Detalhamento da quebra (Negrito e em Azul)

## **TREPORT**

Classe de impressão que substitui as funções SETPRINT, SETDEFAULT, RPTSTATUS e Cabec. Ela 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.

Exemplo

```
#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
```

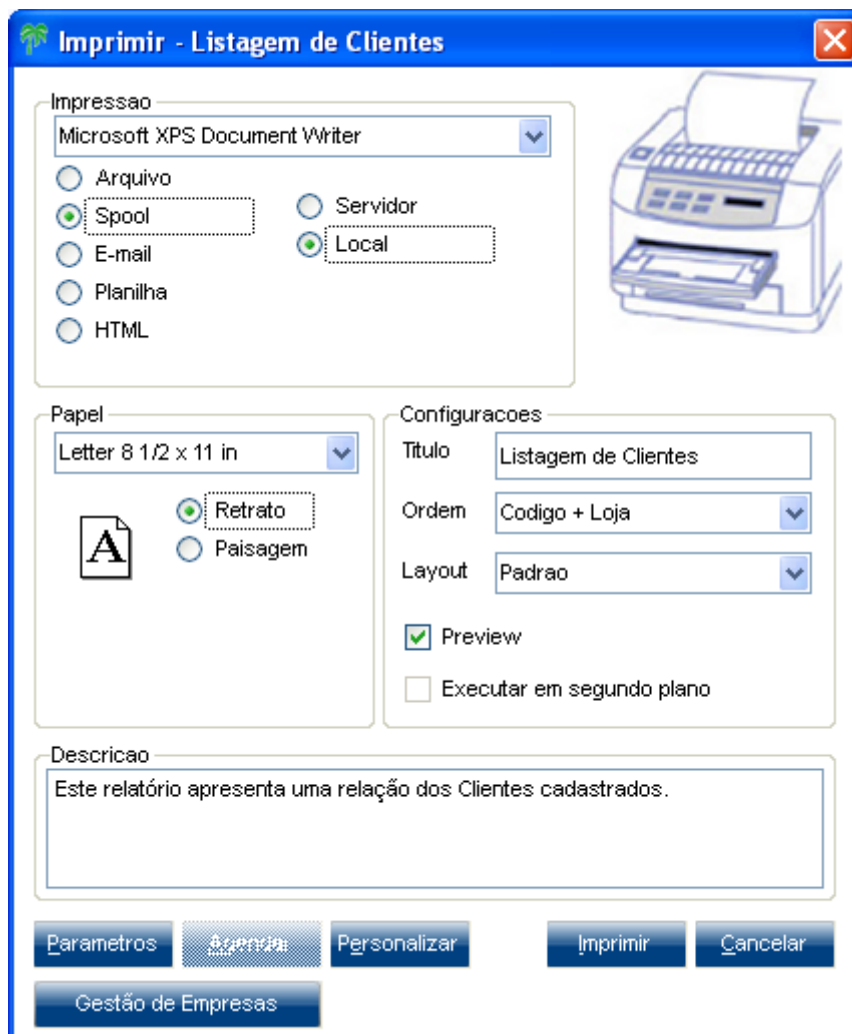
Verifique também o parâmetro MV\_TReport. Para utilizar os relatórios personalizáveis, o parâmetro MV\_TREPORT (tipo numérico) deve ser alterado no ambiente Configurador, conforme uma das opções que seguem:

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

## 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 - Listagem de Clientes**

Impressao  
Microsoft XPS Document Writer

☐ Arquivo  
☒ Spool  
☐ E-mail  
☐ Planilha  
☐ HTML

☐ Servidor  
☒ Local

Papel  
Letter 8 1/2 x 11 in

☒ Retrato  
☐ Paisagem

Configuracoes  
 Titulo: Listagem de Clientes  
 Ordem:Codigo + Loja  
 Layout:Padrao  
☒ Preview  
☐ Executar em segundo plano

Descricao  
Este relatório apresenta uma relação dos Clientes cadastrados.

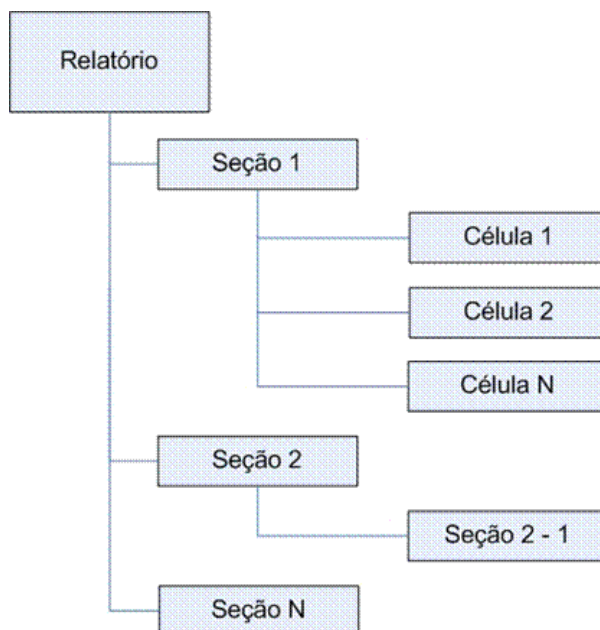
Parametros    Agência    Personalizar    Imprimir    Cancelar  
 Gestão de Empresas

### 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.

### Nova estrutura do relatório TREPORT:



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

### 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:

```

DEFINE REPORT
DEFINE SECTION
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 (TRReport) 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 ().

### Métodos:

<b>New</b>	Método construtor da classe TReport.
<b>SetTitle</b>	Define o título do componente.
<b>SetParam</b>	Define os parametros que serão utilizados pelo usuário para configurar o relatorio.
<b>Preview</b>	Apresenta preview do relatório.
<b>SetAction</b>	Define o bloco de código que será executado quando o usuário confirmar a impressão
<b>Print</b>	Inicia impressão do relatório executando o parametro bAction informado nos métodos New, SetAction ou PrintDialog. Esse método é executado pelo método PrintDialog quando o usuário confirma a impressão do relatório.
<b>PrintDialog</b>	Apresenta a tela de impressão para o usuário configurar o relatório. O método PrintDialog executa o método Print quando o usuário confirma a impressão, executando o bloco de código informado pelo parametro bAction nos métodos New ou SetAction.
<b>Cancel</b>	Retorna se o usuário cancelou a impressão do relatório.

<b>SetMeter</b>	Define o total da regua da tela de processamento do relatório.
<b>IncMeter</b>	Incrementa a régua da tela de processamento do relatório
<b>SetDescription</b>	Define a descrição do componente.
<b>ClassName</b>	Retorna o nome da classe do componente. Exemplo: oReport := TReport():New('MATR010') MsgStop(oReport:ClassName()) //Apresenta na tela 'TREPORT'
<b>ReportName</b>	Retorna o nome do relatório (exemplo: MATR010)
<b>Title</b>	Retorna o título do componente
<b>Description</b>	Descrição do componente
<b>TotalText</b>	Retorna o texto que será impresso antes da impressão dos totalizadores
<b>GetParam</b>	Retorna a pergunta/bloco de código utilizada como parametro do relatório.
<b>GetAction</b>	Retorna bloco de código que será executado pelos métodos Print e PrintDialog quando usuário confirmar impressão.
<b>FooterAtEnd</b>	Retorna se será impressore rodapé no final da impressão
<b>TotalInLine</b>	Retorna se a impressão dos totalizadores será em linha
<b>PageHeight</b>	Retorna a altura da página baseado no tamanho da folha (A4, Carta, etc) informado pelo usuário
<b>PageWidth</b>	Retorna a largura da página baseado no tamanho da folha (A4, Carta, etc) informado pelo usuário
<b>Row</b>	Retorna a linha posicionada na impressão
<b>Col</b>	Retorna a coluna posicionada na impressão
<b>Page</b>	Retorna o número da página atual na impressão
<b>GetOrientation</b>	Retorna a orientação de página (Retrato ou Paisagem) selecionada pelo usuário.
<b>Printing</b>	Retorna se relatório esta em processo de impressão. O status de impressão é definido pelos métodos Init (que indica o início da impressão) e o método Finish (que indica a finalização).
<b>Init</b>	Incializa impressão. Não é necessário executar o método Init se for utilizar o método Print ou PrintDialog, já que estes fazem o controle de inicialização e finalização da impressão. Caso não esteja utilizando os método citados acima, sempre

	execute o método Init no início do relatório para verificar as configurações informadas pelo usuário e inicializar corretamente a primeira página do relatório.
<b>Finish</b>	Finaliza impressão inicializada pelo método Init. Não é necessário executar o método Finish se for utilizar o método Print ou PrintDialog, já que estes fazem o controle de inicialização e finalização da impressão. Caso não esteja utilizando os métodos citados acima, sempre execute o método Finish no final do relatório para impressão dos totalizadores que estão habilitados para o final do relatório, para fechar as queries e/ou índices temporários criados pelas seções e para outros tratamentos internos do componente.
<b>PrintText</b>	Imprime o texto nas coordenadas informadas pelos parâmetros
<b>EndPage</b>	Finaliza uma página de impressão
<b>IncRow</b>	Incrementa a linha de impressão baseado no parâmetro ou no valor definido pelo usuário para a altura das linhas do relatório.
<b>SkipLine</b>	Salta uma linha baseado na altura da linha informada pelo usuário
<b>SetTotalText</b>	Define o texto que será impresso antes da impressão dos totalizadores
<b>SetTotalInLine</b>	Define se os totalizadores serão impressos em linha ou coluna
<b>SetPortrait</b>	Define orientação de página do relatório como retrato
<b>SetLandscape</b>	Define a orientação de página do relatório como paisagem
<b>SetPreview</b>	Define se será apresentado o preview do relatório no final da impressão
<b>GetFunction</b>	Retorna objeto da classe TRFunction (totalizadores) baseado no parâmetro informado.
<b>Section</b>	Retorna objeto da classe TRSection (seção) baseado no parâmetro informado.
<b>Say</b>	Imprime texto nas coordenadas informadas pelos parâmetros, utilizando as especificações do objeto TFont, cor da fonte e cor de fundo
<b>SayBitmap</b>	Imprime uma imagem nas coordenadas informadas pelos parâmetros
<b>Line</b>	Desenha uma linha baseado nas coordenadas informadas pelos parâmetros, utilizando as especificações do objeto TPen.
<b>Box</b>	Desenha uma caixa baseado nas coordenadas informadas pelos parâmetros, utilizando as especificações do objeto TPen.
<b>FillRect</b>	Preenche um retângulo na impressão utilizando as especificações

	do objeto TBrush.
<b>ThinLine</b>	Desenha uma linha simples iniciando na linha atual de impressão com a largura da página
<b>FatLine</b>	Desenha uma linha com altura maior que a do método ThinLine, iniciando na linha atual de impressão com a largura da página

**Exemplo**

```
#include "protheus.ch"
User Function TREP01()

Local oReport

If TRepInUse()
  Pergunte("MTR530",.F.)

  oReport := ReportDef()
  oReport:PrintDialog()
EndIf
Return
```

**Continua...****Pacote office****Integração Excel**

A funcionalidade básica de exportação de informações para o Microsoft Excel utiliza a função DlgToExcel(), a qual permite que as informações em formatos de array sejam geradas em uma planilha.



## DLGTOEXCEL()

Realiza a exportação das informações do ambiente Protheus em formato de arrays para uma planilha do Microsoft Excel.

**Sintaxe:** DLGTOEXCEL( aArray)

aArray := Array multidimensional com 4 elementos, sendo eles:

Argumento	Obrigat.	Tipo	Descrição
cOrigem	Sim	C	Define o que se quer inserir no Excel. 1. CABEÇALHO – Insere a parte superior no Excel 2. GETDADOS – Insere diversos itens 3. ARRAY – Insere um array qualquer
cTitulo	Sim	C	Titulo do Relatorio
aDadosCab	Sim	A	Array contendo os nomes dos campos que serão exibidos na planilha., se cOrigem for GETDADOS, deve ser um array com a estrutura do aHeader.
aDadosItens	Sim	A	Array contendo as informações dos campos, de acordo com a ordem do array de cabeçalho.

### Exemplo

User Function TExcel()

Local aHeader := {}

Local aCols := {}

Local cNum := "000006"

If !APOLECLIENT("MSEExcel")

MsgAlert("Microsoft Excel não instalado!")

Return

EndIf

dbSelectArea("SX3")

dbSetOrder(1)

dbSeek("SC6")

**Continua...**

**Continuação...**

```
While SX3->X3_Arquivo == "SC6" .And. !SX3->(EOF())

  If X3Uso(SX3->X3_Usado) .And.;
    cNivel >= SX3->X3_Nivel .And.;
    Trim(SX3->X3_Campo) $
    "C6_ITEM/C6_PRODUTO/C6_DESCRI/C6_QTDVEN/C6_VALOR"

    AAdd(aHeader, {Trim(SX3->X3_Titulo),;
                  SX3->X3_Campo      ;;
                  SX3->X3_Picture    ;;
                  SX3->X3_Tamanho    ;;
                  SX3->X3_Decimal    ;;
                  SX3->X3_Valid      ;;
                  SX3->X3_Usado      ;;
                  SX3->X3_Tipo       ;;
                  SX3->X3_Arquivo    ;;
                  SX3->X3_Context})

  EndIf

  SX3->(dbSkip())

End

dbSelectArea("SC6")
dbSetOrder(1)
If !dbSeek(xFilial("SC6") + cNum)
  MsgAlert("Pedido nao encontrado!")
  Return
EndIf

While Trim(SC6->C6_Num) == cNum .And. !SC6->(Eof())
  AAdd(aCols, {SC6->C6_ITEM, SC6->C6_PRODUTO, SC6->C6_DESCRI,;
              SC6->C6_QTDVEN, SC6->C6_VALOR, .F.})
  SC6->(dbSkip())
End

dbSelectArea("SC5")
dbSetOrder(1)
If !dbSeek(xFilial("SC5") + cNum)
  MsgAlert("Pedido nao encontrado!")
  Return
EndIf
```

```

Acab := {"Numero","Cliente","Emissao"}
aItem := {SC5->C5_Num,SC5->C5_Cliente,SC5->C5_Emissao}

DlgToExcel({ {"CABECALHO", "Pedido Vendas", aCab, },,
             {"GETDADOS", "Itens do Pedido", aHeader, aCols} })

Return

User Function TExcel2()

Local aCabec := {}
Local aDados := {}

If !ApOleClient("MSEExcel")
    MsgAlert("Microsoft Excel não instalado!")
    Return
EndIf

aCabec := {"Codigo", "Nome", "Endereco"}

dbSelectArea("SA1")
dbGoTop()

While !SA1->(Eof())

    AAdd(aDados, {SA1->A1_Cod, SA1->A1_Nome, SA1->A1_End})

    SA1->(dbSkip())

End

DlgToExcel({ {"ARRAY", "Exportacao para o Excel", aCabec, aDados} })

Return
    
```

### Usando APOLECLIENT()

Podemos usar a Classe MSEExcel para criar integração com Excel.

<b>ApOleClient</b>	Verifica se consegue fazer integração
<b>New</b>	Cria o Objeto de Integração com Excel
<b>Destroy</b>	Destroi o objeto
<b>WorkBooks:Open</b>	Abre uma planilha pronta
<b>SetVisible</b>	Habilita visibilidade

## Exemplo com Arquivo DBF

```
User function fExcDBF()  
  If !ApOleClient("MSEExcel")  
    MsgAlert("Microsoft Excel não instalado!")  
    Return  
  EndIf  
  
  cArq := "SA1990"  
  cPath := GetSrvProfString("ROOTPATH", "C:\MP8") + "\DATA\  
  
  oExcel := MSEExcel():New()  
  oExcel:WorkBooks:Open(cPath + cArq + ".DBF")  
Return
```

## Exemplo Arquivo CSV

```
User Function TExcCSV()  
  
Local oExcel, cArq, nArq, cPath  
  
If !ApOleClient("MSEExcel")  
  MsgAlert("Microsoft Excel não instalado!")  
  Return  
EndIf  
  
cArq := CriaTrab(Nil, .F.)  
cPath := GetSrvProfString("ROOTPATH", "C:\") + "\DIRDOC\  
nArq := FCreate(cPath + cArq + ".CSV")  
  
If nArq == -1  
  MsgAlert("Nao conseguiu criar o arquivo!")  
  Return  
EndIf  
  
FWrite(nArq, "Codigo;Nome;Endereco" + Chr(13) + Chr(10))  
  
dbSelectArea("SA1")  
dbGoTop()  
While !SA1->(Eof())  
  FWrite(nArq, SA1->A1_Cod + ";" + SA1->A1_Nome + ";" + SA1->A1_End + Chr(13)  
  + Chr(10))  
  SA1->(dbSkip())  
End  
  
FClose(nArq)
```

```
oExcel := MSEXcel():New()  
oExcel:WorkBooks:Open(cPath + cArq + ".CSV")  
  
FErase(cPath + cArq + ".CSV")  
  
Return
```

## **Exercício Excel**

Fazer uma consulta usando instrução em SQL e com o resultado da consulta gerar integração com o Excel.

Campos: Num. Pedido , Nome Cliente, Estado, Condição Pgto

Tabelas de Apoio: SC5 e SA1

Perguntar ao usuário se ele prefere resultado em DBF ou CSV.

## **Integração com Word**

**OLE\_CreateLink()** – Cria um objeto de integração com Word

Para criar integração com Word é necessário criar um documento modelo (.DOT) no Word.

Para tornar o documento dinâmico é necessário incluir variáveis nesse modelo:

- Clique no ícone "Inserir um Campo"
- Escolha a opção DOCVARIABLE
- Insira o nome da variável
- Salve o documento como Modelo do Word.

### **Métodos:**

OLE_CreateLink	Conecta ao word
OLE_NewFile	Escolhe o modelo, deve estar visível na máquina do cliente
OLE_SetDocumentVar	Montagem das variáveis que serão lidas no Modelo DOT
OLE_ExecuteMacro	Executa Macro no word
OLE_UpdateFields	Atualizando as variáveis do documento do Word
OLE_PrintFile	Imprime no word

OLE_CloseFile	Fecha o arquivo no word
OLE_CloseLink	Fecha o link no word

#### Exemplo

```
#INCLUDE "PROTHEUS.CH"
USER FUNCTION fCli()
  oWord:=OLE_CreateLink()
  SA1->(dbGotop())
  While SA1->(!EOF())
    OLE_NewFile(oWord, "c:\cli02.dot")
    OLE_SetDocumentVar(oWord,"A1_NOME", SA1->A1_NOME)
    OLE_SetDocumentVar(oWord,"A1_MUN", SA1->A1_MUN)
    OLE_SetDocumentVar(oWord,"A1_COD", SA1->A1_COD)
    OLE_UpdateFields(oWord)
    SA1->(dbSkip())
  Enddo
  If msgyesno("Fecha?")
    OLE_CloseFile(oWord)
    OLE_CloseLink(oWord)
  Endif
Return
```