



# Protheus<sup>10</sup>

---

***Educação Corporativa***

---

## **ADVPL Avançado (Capacitação Interna)**

# ESTRUTURA DO TREINAMENTO

<b>OBJETIVOS DO CURSO.....</b>	<b>11</b>
<b>MÓDULO 05: Introdução à orientação à objetos .....</b>	<b>12</b>
<b>1. Conceitos de orientação à objetos.....</b>	<b>12</b>
<b>1.1. Definições .....</b>	<b>12</b>
<b>1.2. Conceitos Básicos .....</b>	<b>15</b>
<b>1.3. O Modelo de Objetos (OMT) .....</b>	<b>16</b>
1.3.1. Objetos e Classes .....	16
1.3.2. Atributos .....	17
1.3.3. Operações e Métodos .....	18
1.3.4. Sugestões de desenvolvimento.....	19
<b>2. Orientação a objetos em ADVPL .....</b>	<b>20</b>
<b>2.1. Sintaxe e operadores para orientação a objetos .....</b>	<b>20</b>
<b>2.2. Estrutura de uma classe de objetos em ADVPL .....</b>	<b>23</b>
<b>2.3. Implementação dos métodos de uma classe em ADVPL.....</b>	<b>24</b>
<b>3. Regras adicionais da linguagem ADVPL.....</b>	<b>28</b>
<b>3.1. Palavras reservadas.....</b>	<b>28</b>
<b>3.2. Pictures de formatação disponíveis .....</b>	<b>29</b>
<b>MÓDULO 07: ADVPL Orientado à objetos II .....</b>	<b>30</b>
<b>4. Componentes da interface visual do ADVPL .....</b>	<b>30</b>
TSRVOBJECT().....	32
TFONT() .....	35
MSDIALOG() .....	36
TDIALOG().....	37
TWINDOW() .....	40
TCONTROL() .....	43
BRGETDDB() .....	44
MSCALEND() .....	47
MSCALENDGRID() .....	49
MSSELBR() .....	51
MSWORKTIME().....	54
SBUTTON() .....	57
TBAR() .....	59
TBITMAP() .....	61
TBROWSEBUTTON() .....	63
TBTNBMP() .....	65
TBTNBMP2().....	67
TBUTTON() .....	69
TCBROWSE() .....	71
TCHECKBOX() .....	75
TCOLORTRIANGLE() .....	77
TCOMBOBOX() .....	79
TFOLDER() .....	82
TGET() .....	84
TGROUP() .....	86
THBUTTON() .....	88
TIBROWSER() .....	89
TLISTBOX().....	91
TMENU().....	95
TMENUBAR() .....	96
TMETER() .....	97
TMSGGRAPHIC() .....	100
TMSGBAR() .....	106
TMSGITEM().....	108
TMULTIBTN() .....	109
TMULTIGET().....	112
TOLECONTAINER() .....	114

TPAGEVIEW()	116
TPANEL()	118
TRADMENU()	119
TSBROWSE()	123
TSAY()	127
TSCROLLBOX()	129
TSIMPLEEDITOR()	130
TSLIDER()	135
TSPLITTER()	138
TTABS()	142
TTOOLBOX()	144
TWBROWSE()	147
VCBROWSE()	151
<b>4.1. Particularidades dos componentes visuais.....</b>	<b>157</b>
4.1.1. Configurando as cores para os componentes .....	157
<b>5. Aplicações com a interface visual do ADVPL.....</b>	<b>159</b>
<b>5.1. Captura de informações simples (Multi-Gets) .....</b>	<b>159</b>
5.1.1. Enchoice() .....	160
5.1.2. MsMGet().....	162
<b>5.2. Captura de múltiplas informações (Multi-Lines) .....</b>	<b>164</b>
5.2.1. MsGetDB().....	165
5.2.2. MsGetDados() .....	169
5.2.3. MsNewGetDados() .....	173
5.2.3.1. Definindo cores personalizadas para o objeto MsNewGetDados() .....	178
<b>5.3. Barras de botões.....</b>	<b>183</b>
5.3.1. EnchoiceBar().....	183
5.3.2. TBar() .....	185
5.3.3. ButtonBar .....	186
5.3.4. Imagens pré-definidas para as barras de botões.....	189
<b>6. Outras aplicações da interface visual do ADVPL.....</b>	<b>190</b>
<b>6.1. MaWndBrowse().....</b>	<b>190</b>
6.1.1. Enchoice para Arquivos Temporários .....	193
<b>6.2. DbTree() .....</b>	<b>203</b>
<b>6.3. MsSelect() .....</b>	<b>208</b>
<b>7. Introdução à relatórios gráficos.....</b>	<b>211</b>
<b>7.1. TReport().....</b>	<b>211</b>
7.1.1. Introdução .....	211
Finalidade .....	211
Descrição.....	212
Pré-Requisitos .....	212
7.1.2. Impressão do relatório personalizável.....	213
7.1.2.1. Parâmetros de impressão .....	213
Impressão.....	213
Arquivo.....	213
Spool .....	214
E-mail .....	214
Papel.....	215
Tamanho do papel .....	215
Formato da impressão.....	215
Configurações.....	215
Título .....	215
Ordem.....	215
Layout.....	215
Preview .....	215
Executar em segundo plano .....	216
7.1.3. Personalização.....	216
7.1.3.1. Editando o layout do relatório .....	216
Nova estrutura do relatório TReport: .....	217
7.1.4. Definindo a Função ReportDef() .....	217
DEFINE REPORT.....	218
DEFINE SECTION .....	218
DEFINE CELL .....	218

<b>MÓDULO 08: Aplicações ADVPL para o ERP .....</b>	<b>219</b>
<b>8. Protheus e o TOPCONNECT / TOTVS DbAccess .....</b>	<b>219</b>
<b>8.1. Características técnicas com o uso do TOTVS DbAccess.....</b>	<b>220</b>
Comportamento de Queries – Colunas Calculadas .....	220
Comportamento diferenciado com Bandos de Dados PostGres .....	222
Conceito de Índices Permanentes e Diferenças das RDDs .....	223
Funcionamento Interno.....	223
Quebra de Compatibilidade com CodeBase/DBF .....	224
Lista dos códigos de erro do TOPConnect / DbAccess.....	225
<b>8.2. Funções ADVPL para TOPCONNECT / TOTVS DbAccess .....</b>	<b>226</b>
Lista das funções de interação com a aplicação TopConnect / DbAccess:.....	226
Lista das funções acessórias utilizadas nos fontes como facilitadoras:.....	226
Funções de interação com o TopConnect / DbAccess .....	227
TCCANOPEN () .....	227
TCCONTYPE() .....	227
TCDELFILE().....	228
TCGENQRY() .....	229
TCGETDB() .....	229
TCLINK() .....	230
TCQUERY() .....	232
TCQUIT().....	233
TCSETCONN() .....	233
TCSETFIELD() .....	234
TCSPEXEC() .....	236
TCSPEXIST() .....	238
TCSQLERROR().....	238
TCSQLEXEC() .....	239
TCSRVTYP() .....	239
TCUNLINK() .....	240
TCCHKOBJ() .....	241
TCEXEERROR() .....	241
TCPGMEXE() .....	242
TCSYSEX() .....	242
Funções acessórias para uso em fontes com interação com o TOTVS DbAccess.....	243
CHANGEQUERY() .....	243
RETFULLNAME().....	244
RETSQLCOND() .....	245
RETSQLNAME().....	245
RETSQLTABLE() .....	246
SQLCOPY().....	246
SQLORDER() .....	247
SQLTOTRB().....	248
<b>8.3. Aplicações com controle de comunicação com o Banco de Dados.....</b>	<b>249</b>
8.3.1. MaWndBrowse com Alias Temporário gerado por Query .....	249
Exemplo: MaWndBrowse com Alias Temporário gerado por Query .....	250
8.3.2. Banco de dados de interface .....	261
Considerações relevantes sobre as funções TCLink() e TCSetConn() .....	261
Considerações complementares sobre o conceito de Banco de Dados de Interface .....	263
<b>8.4. Embedded SQL – Facilitador de Query's.....</b>	<b>264</b>
Disponibilidade do Recurso.....	264
Características operacionais - Sintaxe.....	265
Limitação:.....	265
Erros de Compilação.....	266
Erros de Execução.....	266
Características operacionais - depuração.....	267
Função auxiliar - GETLastQuery().....	267
<b>9. Funcionalidade MsExecAuto .....</b>	<b>269</b>
Sobre a MsExecAuto e Rotinas Automáticas.....	269
Quando utilizar a funcionalidade MsExecAuto ? .....	270
Processos da aplicação ERP com tratamentos para execução por MsExecAuto.....	270
Quando não utilizar a funcionalidade MsExecAuto .....	271
<b>9.1. Desenvolvendo aplicações com MsExecAuto .....</b>	<b>272</b>
Estrutura de uma rotina com execução de MsExecAuto .....	272
Variáveis de controle .....	272
Montagem dos arrays de parâmetros.....	272
Definição dos parâmetros específicos da rotina que será executada.....	276

Controle de transação .....	277
Tratamento de mensagens de erro.....	279
Aplicação completa de importação utilizando MsExecAuto: Carga de imobilizado .....	282
<b>10. Recursos de envio de e-mail .....</b>	<b>295</b>
Funções para manipulação de e-mails .....	295
Detalhamento das funções de manipulação de e-mails.....	295
CALLPROC() .....	295
MAILSMTPON() .....	296
MAILPOPON().....	296
MAILSMTPOFF().....	297
MAILPOPOFF().....	297
MAILRECEIVE().....	298
MAILAUTH() .....	299
POPMSGCOUNT().....	300
MAILSEND().....	300
MAILGETERR().....	301
Exemplos de utilização das funcionalidades de envio e recebimento de e-mail .....	301
Envio de mensagens utilizando sintaxe clássica .....	301
Envio de mensagens utilizando funções .....	304
Recebimento de mensagens utilizando funções .....	307
<b>11. Integração básica com MsOffice.....</b>	<b>309</b>
<b>11.1. Exportação para EXCEL .....</b>	<b>309</b>
DLGTOEXCEL() .....	309
Exemplo de exportação para o Microsoft Excel utilizando a função DlgToExcel().....	310
<b>APÊNDICES.....</b>	<b>312</b>
<b>Relação de imagens para aplicações visuais.....</b>	<b>312</b>
<b>LISTAS DE EXERCÍCIOS .....</b>	<b>320</b>
<b>Projeto: Avaliação prática do treinamento de ADVPL Avançado .....</b>	<b>325</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>326</b>
<b>GUIA DE REFERÊNCIA RÁPIDA: Funções e Comandos ADVPL .....</b>	<b>328</b>
<b>Conversão entre tipos de dados.....</b>	<b>328</b>
CTOD() .....	328
CVALTOCHAR().....	328
DTOC().....	329
DTOS().....	329
STOD() .....	329
STR() .....	330
STRZERO() .....	330
VAL() .....	331
<b>Matemáticas.....</b>	<b>332</b>
ACOS().....	332
CEILING().....	332
COS().....	332
LOG10() .....	333
SIN().....	333
SQRT().....	334
TAN() .....	334
<b>Análise de variáveis.....</b>	<b>335</b>
TYPE().....	335
VALTYPE().....	335
<b>Manipulação de arrays.....</b>	<b>336</b>
AADD() .....	336
ACLONE() .....	337
ACOPY() .....	337
ADEL() .....	338
ADIR() .....	339
AFILL() .....	340
AINS().....	340
ARRAY() .....	341
ASCAN() .....	341
ASCANX() .....	342
ASIZE() .....	343

ASORT() .....	344
ATAIL() .....	345
<b>Manipulação de blocos de código.....</b>	<b>346</b>
EVAL().....	346
DBEVAL().....	346
AEVAL().....	348
<b>Manipulação de strings.....</b>	<b>349</b>
ALLTRIM().....	349
ASC() .....	349
AT() .....	350
BITON().....	351
CAPITAL().....	351
CHR().....	351
DESCEND() .....	352
GETDTOVAL().....	352
ISALPHA().....	353
ISDIGIT() .....	353
ISLOWER() .....	354
ISUPPER().....	354
LEN() .....	354
LOWER().....	355
LTRIM() .....	355
MATHC().....	356
OEMTOANSI() .....	356
PADL() / PADR() / PADC().....	357
RAT() .....	357
REPLICATE() .....	358
RETASC().....	358
RTRIM().....	359
SPACE().....	359
STRTOKARR().....	360
STRTRAN().....	360
STUFF() .....	361
SUBSTR() .....	361
TRANSFORM() .....	362
UPPER().....	362
<b>Manipulação de data / hora.....</b>	<b>363</b>
CDOW().....	363
CMONTH() .....	363
DATE() .....	364
DAY() .....	364
DOW().....	365
DTOC().....	365
DTOS().....	366
ELAPTIME() .....	366
MONTH() .....	367
SECONDS() .....	367
TIME().....	368
YEAR() .....	368
<b>Manipulação de variáveis numéricas .....</b>	<b>369</b>
ABS() .....	369
ALEATORIO() .....	369
INT() .....	370
NOROUND() .....	370
RANDOMIZE() .....	371
ROUND() .....	371
<b>Manipulação de arquivos .....</b>	<b>372</b>
ADIR().....	372
CGETFILE() .....	373
CPYS2T() .....	379
CPYT2S() .....	379
CURDIR() .....	380
DIRECTORY() .....	381
DIRREMOVE() .....	382
DISKSPACE() .....	383
EXISTDIR() .....	384
FCLOSE().....	384

FCREATE()	385
FERASE()	386
FILE()	386
FILENOEXT()	387
FOPEN()	388
FREAD()	390
FREADSTR()	390
FRENAME()	391
FSEEK()	392
FT_FEOF()	392
FT_FGOTO()	393
FT_FGOTOP()	393
FT_FLASTREC()	393
FT_FREADLN()	394
FT_FRECNO()	395
FT_FSKIP()	395
FT_FUSE()	395
FWRITE()	396
MSCOPYFILE()	398
MSCOPYTO()	399
MSCREATE()	399
MSERASE()	400
MSRENAME()	401
RETFilename()	401
<b>Manipulação de arquivos e índices temporários</b>	<b>402</b>
CRIATRAB()	402
<b>Manipulação de bases de dados</b>	<b>403</b>
ALIAS()	403
BOF() / EOF()	403
COPY()	404
COPY STRUCTURE()	407
DBAPPEND()	407
DBCLEARALLFILTER()	408
DBCLEARFILTER()	409
DBCLEARINDEX()	409
DBCLOSEALL()	410
DBCLOSEAREA()	410
DBCOMMIT()	411
DBCOMMITALL()	411
DBCREATE()	412
DBCREATEINDEX()	413
DBDELETE()	414
DBF()	415
DBFIELDINFO()	415
DBFILTER()	416
DBGOTO()	417
DBGOTOP()	417
DBGOBUTTON()	418
DBINFO()	418
DBNICKINDEXKEY()	419
DBORDERINFO()	420
DBORDERNICKNAME()	421
DBPACK()	421
DBRECALL()	422
DBRECORDINFO()	422
DBREINDEX()	423
DBRLOCK()	424
DBRLOCKLIST()	424
DBRUNLOCK()	425
DBSETDRIVER()	425
DBSETINDEX()	426
DBSETNICKNAME()	427
DBSELECTAREA()	427
DBSETORDER()	428
DBSEEK() E MSSEEK()	429
DBSKIP()	430
DBSETFILTER()	431
DBSTRUCT()	432
DBUNLOCK()	432

DBUNLOCKALL()	433
DBUSEAREA()	433
DELETED()	434
FCOUNT()	434
FOUND()	435
INDEXKEY()	435
INDEXORD()	436
LUPDATE()	436
MSAPPEND()	437
MSUNLOCK()	437
ORDBAGEXT()	438
ORDKEY()	438
RECLOCK()	439
RECNO()	440
SELECT()	440
SET FILTER TO	441
SOFTLOCK()	442
USED()	443
ZAP	443
<b>Controle de numeração seqüencial.</b>	<b>444</b>
GETSXENUM()	444
CONFIRMSXE()	444
ROLLBACKSXE()	444
<b>Validação.</b>	<b>445</b>
ALLWAYSFALSE()	445
ALLWAYSTRUE()	445
EXISTCHAV()	445
EXISTCPO()	446
LETTERORNUM()	446
NAOVAZIO()	446
NEGATIVO()	446
PERTENCE()	447
POSITIVO()	447
TEXTO()	447
VAZIO()	447
<b>Manipulação de parâmetros do sistema.</b>	<b>448</b>
GETMV()	448
GETNEWPAR()	448
PUTMV()	449
SUPERGETMV()	449
<b>Controle de impressão.</b>	<b>450</b>
AVALIMP()	450
CABEC()	451
IMPCADAST()	454
MS_FLUSH()	454
OURSPOOL()	456
RODA()	457
SETDEFAULT()	459
SETPRC()	460
SETPRINT()	460
<b>Controle de processamentos</b>	<b>462</b>
ABREEXCL()	462
CLOSEOPEN()	462
CLOSESFILE()	462
CHKFILE()	463
CONOUT()	464
CRIAVAR()	464
DISARMTRANSACTION()	465
EXECBLOCK()	466
EXISTBLOCK()	466
ERRORBLOCK()	467
FINAL()	469
FINDFUNCTION()	469
FUNDESC()	470
FUNNAME()	470
GETAREA()	470
GETCOUNTRYLIST()	471

ISINCALLSTACK()	471
REGTOMEMORY()	472
RESTAREA()	473
USEREXCEPTION()	473
<b>Utilização de recursos do ambiente ERP .....</b>	<b>475</b>
AJUSTASX1()	475
ALLUSERS()	477
ALLGROUPS()	479
CGC()	480
CONPAD1()	480
DATAVALIDA()	481
EXISTINI()	481
EXTENSO()	482
FORMULA()	482
GETADVFVAL()	482
HELP()	483
MESEXTENSO()	484
OBIGATORIO()	484
OPENFILE()	487
PERGUNTE()	488
PESQPICT()	488
PESQPICTQT()	489
POSICIONE()	489
PUTSX1()	490
RETINDEX()	491
SIXDESCRICAO()	491
TABELA()	492
TAMSX3()	492
TM()	493
X1DEF01()	494
X1PERGUNT()	495
X2NOME()	495
X3CBOX()	496
X3DESCRIC()	496
X3PICTURE()	497
X3TITULO()	498
X3USO()	498
X5DESCRI()	499
X6CONTEUD()	500
X6DESCRIC()	501
XADESCRIC()	502
XBDESCRI()	502
XFILIAL()	503
<b>Interfaces de cadastro .....</b>	<b>504</b>
AXCADASTRO()	504
AXPESQUI()	506
AXVISUAL()	506
AXINCLUI()	507
AXALTERA()	508
AXDELETA()	508
BRWLEGENDA()	509
ENDFILBRW()	509
FILBROWSE()	510
PESQBRW()	510
MARKBROW()	511
MBROWSE()	515
MODELO2()	523
MODELO3()	526
<b>Interfaces visuais para aplicações.....</b>	<b>529</b>
ALERT()	529
AVISO()	530
FORMBACTH()	530
MSGFUNCTIONS()	531
<b>Recursos das interfaces visuais.....</b>	<b>532</b>
GDFIELDGET()	532
GDFIELDPOS()	532
GDFIELDPUT()	532

GETMARK()	533
ISMARK()	534
MARKBREFRESH()	534
READVAR()	535
THISINV()	535
THISMARK()	535



*Anotações*

---

---

---

---

## **OBJETIVOS DO CURSO**

### **Objetivos específicos do curso:**

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

#### **a) Conceitos a serem aprendidos**

- estruturas para implementação aplicações orientadas a objetos
- introdução as técnicas de programação de desenvolvimento de relatórios gráficos
- introdução aos conceitos de ADVPL ASP e WebServices

#### **b) Habilidades e técnicas a serem aprendidas**

- desenvolvimento de aplicações voltadas ao ERP Protheus
- análise de fontes de média complexidade
- desenvolvimento com orientação a objetos

#### **c) Atitudes a serem desenvolvidas**

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

# MÓDULO 05: Introdução à orientação à objetos

## 1. Conceitos de orientação à objetos

O termo orientação a objetos pressupõe uma organização de software em termos de coleção de objetos discretos incorporando estrutura e comportamento próprios. Esta abordagem de organização é essencialmente diferente do desenvolvimento tradicional de software, onde estruturas de dados e rotinas são desenvolvidas de forma apenas fracamente acopladas.

Neste tópico serão os conceitos de programação orientada a objetos listados abaixo. Esta breve visão geral do paradigma permitirá entender melhor os conceitos associados à programação orientada a objetos e, em particular, às construções implementadas através da linguagem ADVPL.

- Objetos**
- Herança**
- Atributos**
- Métodos**
- Classes**
- Abstração**
- Generalização**
- Encapsulamento**
- Polimorfismo**

### 1.1. Definições

#### Objeto

Um objeto é uma entidade do mundo real que tem uma identidade. Objetos podem representar entidades concretas (um arquivo no meu computador, uma bicicleta) ou entidades conceituais (uma estratégia de jogo, uma política de escalonamento em um sistema operacional). Cada objeto ter sua identidade significa que dois objetos são distintos mesmo que eles apresentem exatamente as mesmas características.

Embora objetos tenham existência própria no mundo real, em termos de linguagem de programação um objeto necessita um mecanismo de identificação. Esta identificação de objeto deve ser única, uniforme e independente do conteúdo do objeto. Este é um dos mecanismos que permite a criação de coleções de objetos, as quais são também objetos em si.

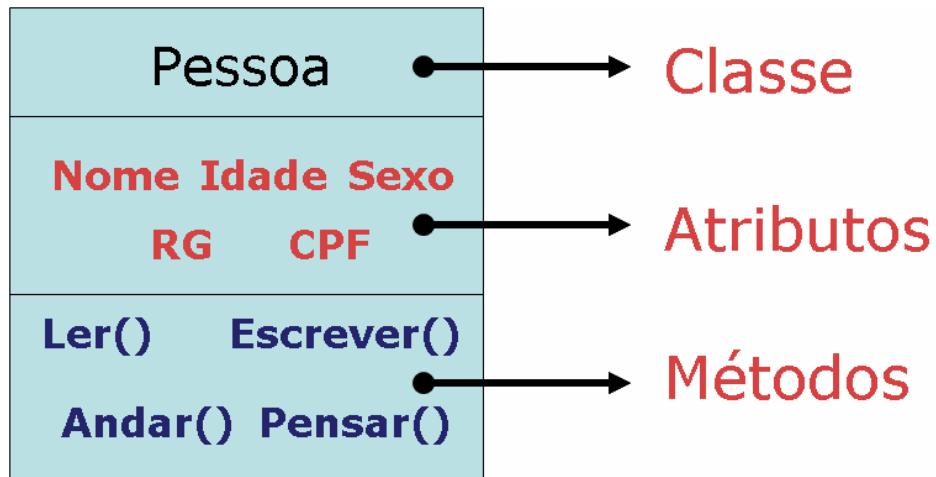
A estrutura de um objeto é representada em termos de atributos. O comportamento de um objeto é representado pelo conjunto de operações que podem ser executadas sobre o objeto.

## Classe

---

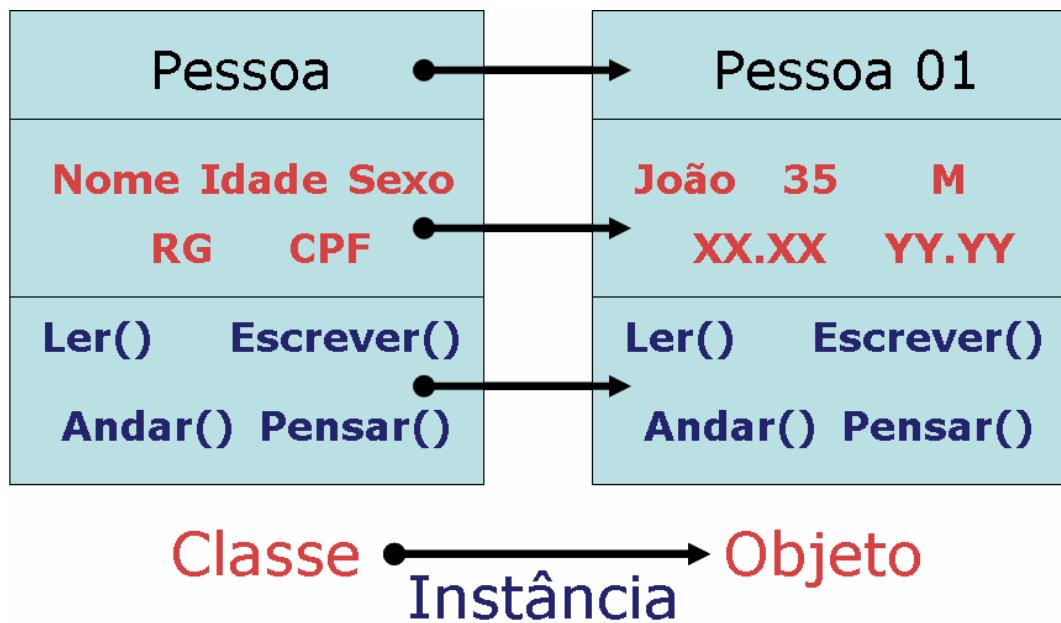
Objetos com a mesma estrutura e o mesmo comportamento são agrupados em classes. Uma classe é uma abstração que descreve propriedades importantes para uma aplicação e simplesmente ignora o resto.

Cada classe descreve um conjunto (possivelmente infinito) de objetos individuais. Cada objeto é dito ser uma instância de uma classe. Assim, cada instância de uma classe tem seus próprios valores para cada atributo, mas dividem os nomes dos atributos e métodos com as outras instâncias da classe. Implicitamente, cada objeto contém uma referência para sua própria classe, em outras palavras, ele sabe o que ele é.



**Figura:** Representação de uma classe de objetos

---



**Figura:** Representação de um objeto instanciado de uma classe

---

## **Polimorfismo**

---

Polimorfismo significa que a mesma operação pode se comportar de forma diferente em classes diferentes. Por exemplo, a operação move quando aplicada a uma janela de um sistema de interfaces tem um comportamento distinto do que quando aplicada a uma peça de um jogo de xadrez. Um método é uma implementação específica de uma operação para uma certa classe.

Polimorfismo também implica que uma operação de uma mesma classe pode ser implementada por mais de um método. O usuário não precisa saber quantas implementações existem para uma operação, ou explicitar qual método deve ser utilizado: a linguagem de programação deve ser capaz de selecionar o método correto a partir do nome da operação, classe do objeto e argumentos para a operação. Desta forma, novas classes podem ser adicionadas sem necessidade de modificação de código já existente, pois cada classe apenas define os seus métodos e atributos.

No mundo real, alguns objetos e classes podem ser descritos como casos especiais, ou especializações, de outros objetos e classes. Por exemplo, a classe de computadores pessoais com processador da linha 80x86 é uma especialização de computadores pessoais, que por sua vez é uma especialização de computadores. Não é desejável que tudo que já foi descrito para computadores tenha de ser repetido para computadores pessoais ou para computadores pessoais com processador da linha 80x86.

## **Herança**

---

Herança é o mecanismo do paradigma de orientação a objetos que permite compartilhar atributos e operações entre classes baseada em um relacionamento hierárquico. Uma classe pode ser definida de forma genérica e depois refinada sucessivamente em termos de subclasses ou classes derivadas. Cada subclasse incorpora, ou herda, todas as propriedades de sua superclasse (ou classe base) e adiciona suas propriedades únicas e particulares. As propriedades da classe base não precisam ser repetidas em cada classe derivada. Esta capacidade de fatorar as propriedades comuns de diversas classes em uma superclasse pode reduzir dramaticamente a repetição de código em um projeto ou programa, sendo uma das principais vantagens da abordagem de orientação a objetos.

## **1.2. Conceitos Básicos**

---

A abordagem de orientação a objetos favorece a aplicação de diversos conceitos considerados fundamentais para o desenvolvimento de bons programas, tais como abstração e encapsulamento.

Tais conceitos não são exclusivos desta abordagem, mas são suportados de forma melhor no desenvolvimento orientado a objetos do que em outras metodologias.

### **Abstração**

---

Abstração consiste de focalizar nos aspectos essenciais inerentes a uma entidade e ignorar propriedades "acidentais". Em termos de desenvolvimento de sistemas, isto significa concentrar-se no que um objeto é e faz antes de se decidir como ele será implementado. O uso de abstração preserva a liberdade para tomar decisões de desenvolvimento ou de implementação apenas quando há um melhor entendimento do problema a ser resolvido.

Muitas linguagens de programação modernas suportam o conceito de abstração de dados; porém, o uso de abstração juntamente com polimorfismo e herança, como suportado em orientação a objetos, é um mecanismo muito mais poderoso.

O uso apropriado de abstração permite que um mesmo modelo conceitual (orientação a objetos) seja utilizado para todas as fases de desenvolvimento de um sistema, desde sua análise até sua documentação.

### **Encapsulamento**

---

Encapsulamento, também referido como esconder informação, consiste em separar os aspectos externos de um objeto, os quais são acessíveis a outros objetos, dos detalhes internos de implementação do objeto, os quais permanecem escondidos dos outros objetos. O uso de encapsulamento evita que um programa torne-se tão interdependente que uma pequena mudança tenha grandes efeitos colaterais.

O uso de encapsulamento permite que a implementação de um objeto possa ser modificada sem afetar as aplicações que usam este objeto. Motivos para modificar a implementação de um objeto podem ser, por exemplo, melhoria de desempenho, correção de erros e mudança de plataforma de execução.

Assim como abstração, o conceito de Encapsulamento não é exclusivo da abordagem de orientação a objetos. Entretanto, a habilidade de se combinar estrutura de dados e comportamento em uma única entidade torna a Encapsulamento mais elegante e mais poderosa do que em linguagens convencionais que分离 estruturas de dados e comportamento.

### **Compartilhamento**

---

Técnicas de orientação a objetos promovem compartilhamento em diversos níveis distintos. Herança de estrutura de dados e comportamento permite que estruturas comuns sejam compartilhadas entre diversas classes derivadas similares sem redundância. O compartilhamento de código usando herança é uma das grandes vantagens da orientação a objetos. Ainda mais importante que a economia de código é a clareza conceitual de reconhecer que operações diferentes são na verdade a mesma coisa, o que reduz o número de casos distintos que devem ser entendidos e analisados.

O desenvolvimento orientado a objetos não apenas permite que a informação dentro de um projeto seja compartilhada como também oferece a possibilidade de reaproveitar projetos e código em projetos futuros. As ferramentas para alcançar este compartilhamento, tais como abstração, Encapsulamento e herança, estão presentes na metodologia; uma estratégia de reuso entre projetos é a definição de bibliotecas de elementos reusáveis. Entretanto, orientação a objetos não é uma fórmula mágica para alcançar reusabilidade; para tanto, é preciso planejamento e disciplina para pensar em termos genéricos, não voltados simplesmente para a aplicação corrente.

### 1.3. O Modelo de Objetos (OMT)

Um modelo de objetos busca capturar a estrutura estática de um sistema mostrando os objetos existentes, seus relacionamentos, e atributos e operações que caracterizam cada classe de objetos. É através do uso deste modelo que se enfatiza o desenvolvimento em termos de objetos ao invés de mecanismos tradicionais de desenvolvimento baseado em funcionalidades, permitindo uma representação mais próxima do mundo real.

Uma vez que as principais definições e conceitos da abordagem de orientação a objetos estão definidos, é possível introduzir o modelo de objetos que será adotado ao longo deste texto. O modelo apresentado é um subconjunto do modelo OMT (Object Modeling Technique), proposto por Rumbaugh entre outros. Este modelo também introduz uma representação diagramática para este modelo, a qual será também apresentada aqui.

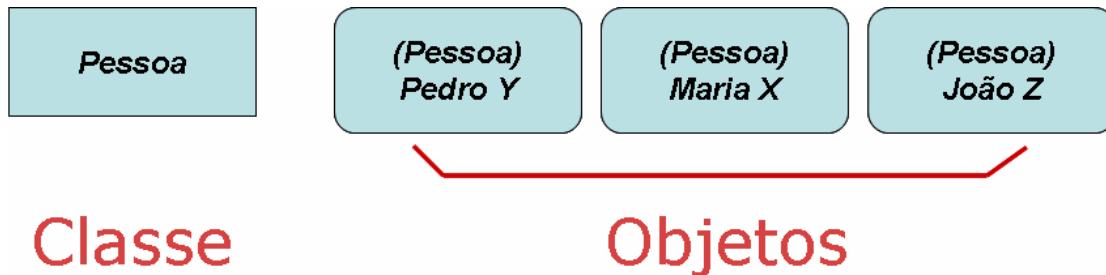
#### 1.3.1. Objetos e Classes

Objeto é definido neste modelo como um conceito, abstração ou coisa com limites e significados bem definidos para a aplicação em questão. Objetos têm dois propósitos: promover o entendimento do mundo real e suportar uma base prática para uma implementação computacional. Não existe uma maneira “correta” de decompor um problema em objetos; esta decomposição depende do julgamento do projetista e da natureza do problema. Todos os objetos têm identidade própria e são distinguíveis.

Uma classe de objetos descreve um grupo de objetos com propriedades (atributos) similares, comportamentos (operações) similares, relacionamentos comuns com outros objetos e uma semântica comum. Por exemplo, Pessoa e Companhia são classes de objetos. Cada pessoa tem um nome e uma idade; estes seriam os atributos comuns da classe. Companhias também podem ter os mesmos atributos nome e idade definidos. Entretanto, devido à distinção semântica elas provavelmente estariam agrupados em outra classe que não Pessoa. Como se pode observar, o agrupamento em classes não leva em conta apenas o compartilhamento de propriedades.

Todo objeto sabe a que classe ele pertence, ou seja, a classe de um objeto é um atributo implícito do objeto. Este conceito é suportado na maior parte das linguagens de programação orientada a objetos, inclusive em ADVPL.

OMT define dois tipos de diagramas de objetos, diagramas de classes e diagramas de instâncias. Um diagrama de classe é um esquema, ou seja, um padrão ou gabarito que descreve as muitas possíveis instâncias de dados. Um diagrama de instâncias descreve como um conjunto particular de objetos está relacionado. Diagramas de instâncias são úteis para apresentar exemplos e documentar casos de testes; diagramas de classes têm uso mais amplo. A Figura abaixo apresenta a notação adotada para estes diagramas.



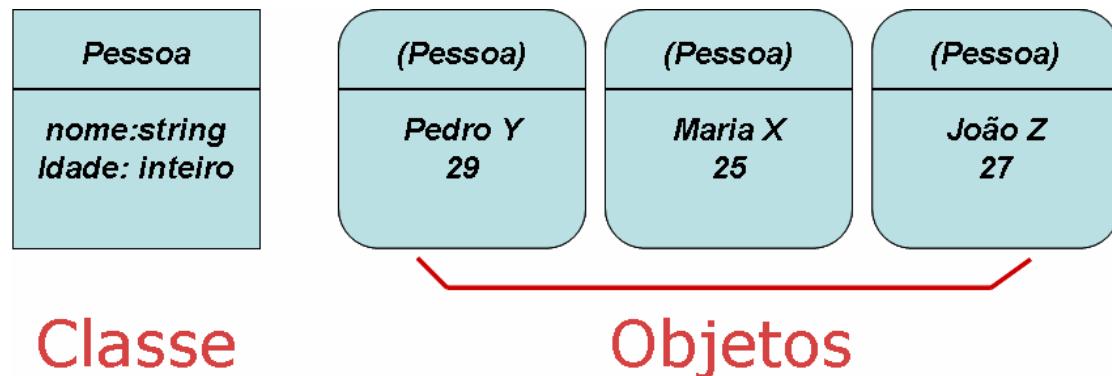
**Figura:** Representação diagramática de OMT para classes e objetos

O agrupamento de objetos em classes é um poderoso mecanismo de abstração. Desta forma, é possível generalizar definições comuns para uma classe de objetos, ao invés de repetí-las para cada objeto em particular. Esta é uma das formas de reutilização e economia que a abordagem de orientação a objetos suporta.

### 1.3.2. Atributos

Um atributo é um valor de dado assumido pelos objetos de uma classe. Nome, idade e peso são exemplos de atributos de objetos Pessoa. Cor, peso e modelo são possíveis atributos de objetos Carro. Cada atributo tem um valor para cada instância de objeto. Por exemplo, o atributo idade tem valor ``29'' no objeto Pedro Y. Em outras palavras, Pedro Y tem 29 anos de idade. Diferentes instâncias de objetos podem ter o mesmo valor para um dado atributo. Cada nome de atributo é único para uma dada classe, mas não necessariamente único entre todas as classes. Por exemplo, ambos Pessoa e Companhia podem ter um atributo chamado endereço.

No diagrama de classes, atributos são listados no segundo segmento da caixa que representa a classe. O nome do atributo pode ser seguido por detalhes opcionais, tais como o tipo de dado assumido e valor default. A Figura abaixo mostra esta representação.



**Figura:** Representação diagramática de OMT para classes e objetos com atributos

Não se deve confundir identificadores internos de objetos com atributos do mundo real. Identificadores de objetos são uma conveniência de implementação, e não têm nenhum significado para o domínio da aplicação. Por exemplo, CIC e RG não são identificadores de objetos, mas sim verdadeiros atributos do mundo real.

### 1.3.3. Operações e Métodos

Uma operação é uma função ou transformação que pode ser aplicada a ou por objetos em uma classe. Por exemplo, abrir, salvar e imprimir são operações que podem ser aplicadas a objetos da classe Arquivo. Todos os objetos em uma classe compartilham as mesmas operações.

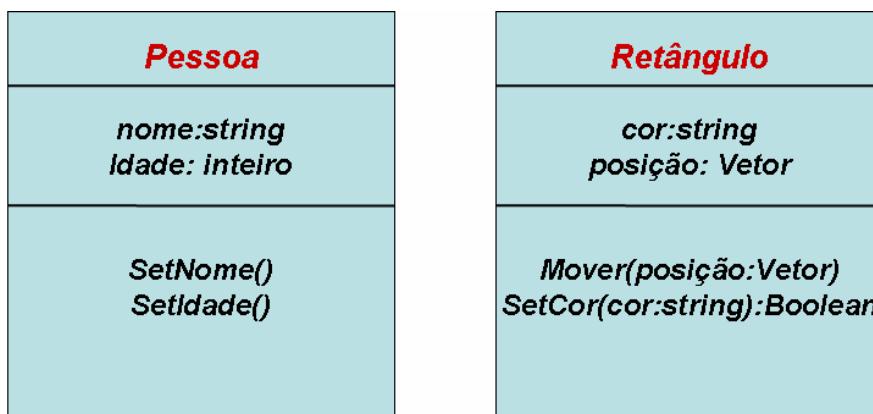
Toda operação tem um objeto-alvo como um argumento implícito. O comportamento de uma operação depende da classe de seu alvo. Como um objeto "sabe" qual sua classe, é possível escolher a implementação correta da operação. Além disto, outros argumentos (parâmetros) podem ser necessários para uma operação.

Uma mesma operação pode se aplicar a diversas classes diferentes. Uma operação como esta é dita ser polimórfica, ou seja, ela pode assumir distintas formas em classes diferentes.

Um método é a implementação de uma operação para uma classe. Por exemplo, a operação imprimir pode ser implementada de forma distinta, dependendo se o arquivo a ser impresso contém apenas texto ASCII, é um arquivo de um processador de texto ou binário. Todos estes métodos executam a mesma operação: imprimir o arquivo; porém, cada método será implementado por um diferente código.

A assinatura de um método é dada pelo número e tipos de argumentos do método, assim como por seu valor de retorno. Uma estratégia de desenvolvimento recomendável é manter assinaturas coerentes para métodos implementando uma dada operação, assim como um comportamento consistente entre as implementações.

Em termos de diagramas OMT, operações são listadas na terceira parte da caixa de uma classe. Cada nome de operação pode ser seguida por detalhes opcionais, tais como lista de argumentos e tipo de retorno. A lista de argumentos é apresentada entre parênteses após o nome da operação. Uma lista de argumentos vazia indica que a operação não tem argumentos; da ausência da lista de argumentos não se pode concluir nada. O tipo de resultado vem após a lista de argumentos, sendo precedido por dois pontos (:). Caso a operação retorne resultado, este não deve ser omitido, pois esta é a forma de distingui-la de operações que não retornam resultado. Exemplos de representação de operações em OMT são apresentados na Figura abaixo:



**Figura:** Representação diagramática de OMT para classes com atributos e operações

#### **1.3.4. Sugestões de desenvolvimento**

Na construção de um modelo para uma aplicação, as seguintes sugestões devem ser observadas a fim de se obter resultados claros e consistentes:

- Não comece a construir um modelo de objetos simplesmente definindo classes, associações e heranças. A primeira coisa a se fazer é entender o problema a ser resolvido.
- Tente manter seu modelo simples. Evite complicações desnecessárias.
- Escolha nomes cuidadosamente. Nomes são importantes e carregam conotações poderosas. Nomes devem ser descritivos, claros e não deixar ambigüidades. A escolha de bons nomes é um dos aspectos mais difíceis da modelagem.
- Não “enterre” apontadores ou outras referências a objetos dentro de objetos como atributos. Ao invés disto, modele estas referências como associações. Isto torna o modelo mais claro e independente da implementação.
- Tente evitar associações que envolvam três ou mais classes de objetos. Muitas vezes, estes tipos de associações podem ser decompostos em termos de associações binárias, tornando o modelo mais claro.
- Não transfira os atributos de ligação para dentro de uma das classes.
- Tente evitar hierarquias de generalização muito profundas.
- Não se surpreenda se o seu modelo necessitar várias revisões; isto é o normal.
- Sempre documente seus modelos de objetos. O diagrama pode especificar a estrutura do modelo, mas nem sempre é suficiente para descrever as razões por trás da definição do modelo. Uma explicação escrita pode clarificar pontos tais como significado de nomes e explicar a razão para cada classe e relacionamento.
- Nem sempre todas as construções OMT são necessárias para descrever uma aplicação. Use apenas aquelas que forem adequadas para o problema analisado.

## **2. Orientação a objetos em ADVPL**

Neste tópico será detalhada a forma com a qual a linguagem ADVPL implementa os conceitos de orientação a objetos e a sintaxe utilizada no desenvolvimento de aplicações.

### **2.1. Sintaxe e operadores para orientação a objetos**

#### **Palavras reservadas**

- CLASS**
- CLASSDATA**
- CONSTRUCTOR**
- DATA**
- ENDCLASS**
- EXPORT**
- FROM**
- HIDDEN**
- METHOD**
- PROTECTED**
- SELF**

#### **CLASS**

<b>Descrição</b>	Utilizada na declaração de uma classe de objetos para identificar a qual classe um determinado método está relacionado.
<b>Sintaxe 1</b>	CLASS <nome_da_classe>
<b>Sintaxe 2</b>	METHOD <nome_do_método> CLASS <nome_da_classe>

#### **CLASSDATA**

<b>Descrição</b>	Utilizada na declaração de um atributo da classe de objetos com a finalidade de definir um tipo e um conteúdo padrão para este atributo.
<b>Sintaxe 1</b>	CLASSDATA <nome_da_variável> AS <tipo> <atributo> INIT <conteúdo>

#### **CONSTRUCTOR**

<b>Descrição</b>	Utilizada na especificação de um método especial, definido como construtor, o qual tem a função de retornar um novo objeto com os atributos e métodos definidos na classe.
<b>Sintaxe</b>	METHOD <nome_do_método(>) CONSTRUCTOR

## **DATA**

---

<b>Descrição</b>	Utilizada na declaração de um atributo da classe de objetos.
<b>Sintaxe</b>	DATA <nome_do_atributo>

## **ENDCLASS**

---

<b>Descrição</b>	Utilizada na finalização da declaração da classe.
<b>Sintaxe</b>	ENDCLASS

## **EXPORT**

---

<b>Descrição</b>	Operador utilizado para definir o modo de acesso de um atributo especificado na classe de objetos. O uso deste operador indica que o atributo será visível e alterável sem a necessidade de utilização dos métodos do próprio objeto. Seu uso na linguagem ADVPL é opcional.
<b>Simular</b>	EXPORT ou [PUBLIC]
<b>Sintaxe 1</b>	DATA cNome
<b>Sintaxe 2</b>	EXPORT DATA cNome

## **FROM**

---

<b>Descrição</b>	Utilizada na declaração de uma classe, a qual será uma instância de uma superclasse, recebendo os atributos e métodos nela definidos, implementando a herança entre classes.
<b>Sintaxe</b>	CLASS <nome_da_classe> FROM <nome_da_superclasse>

## **HIDDEN**

---

<b>Descrição</b>	Operador utilizado para definir o modo de acesso de um atributo especificado na classe de objetos. O uso deste operador indica que o atributo não será visível para chamadas diretas do objeto, sendo necessária a utilização de métodos para sua manipulação.
<b>Simular</b>	HIDDEN ou [LOCAL]
<b>Sintaxe</b>	HIDDEN DATA cNome

## **METHOD**

---

<b>Descrição</b>	Utilizada na declaração do protótipo do método de uma classe de objetos, e na declaração do método efetivamente desenvolvido.
<b>Sintaxe 1</b>	METHOD <nome_do_método(>)
<b>Sintaxe 2</b>	METHOD <nome_do_método(<parâmetros>)> CLASS <nome_da_classe>

## PROTECTED

<b>Descrição</b>	Operador utilizado para definir o modo de acesso de um atributo especificado na classe de objetos. O uso deste operador indica que o atributo será visível mas não será editável para chamadas diretas do objeto, sendo necessária a utilização de métodos para sua manipulação.
<b>Simular</b>	PROTECTED ou [READONLY]
<b>Sintaxe</b>	PROTECTED DATA cNome

## SELF

<b>Descrição</b>	Utilizada principalmente pelo método construtor para retornar o objeto criado para a aplicação.
<b>Sintaxe</b>	Return SELF

## Operadores específicos

:	Utilizado para referenciar um método ou um atributo de um objeto já instanciado.
<b>Exemplo 1</b>	cNome := oAluno: <b>sNome</b>
<b>Exemplo 2</b>	cNota := oAluno: <b>GetNota(cCurso)</b>

::	Utilizado pelos métodos de uma classe para referenciar os atributos disponíveis para o objeto.
<b>Exemplo</b>	METHOD GetNota(cCurso) CLASS ALUNO  Local nPosCurso := 0 Local nNota := 0  nPosCurso := aScan(:::aCursos,{ aCurso  aCurso[1] == cCurso})  IF nPosCurso > 0  nNota := :::aCursos[nPosCurso][2]  ENDIF  Return nNota

## **2.2. Estrutura de uma classe de objetos em ADVPL**

### **Declaração da classe**

A declaração de uma classe da linguagem ADVPL é realizada de forma similar a declaração de uma função, com a diferença de que uma classe não possui diferenciação quanto a sua procedência, como uma Function() e uma User Function(), e não possui visibilidade limitada como uma Static Function().

Exemplo:

```
#include "protheus.ch"
CLASS Pessoa()
```

### **Definição dos atributos**

Seguindo o mesmo princípio de variáveis não tipadas, os atributos das classes em ADVPL não precisam ter seu tipo especificado, sendo necessário apenas determinar seus nomes.

Desta forma é recomendado o uso da notação Húngara também para a definição dos atributos de forma a facilitar a análise, interpretação e utilização da classe e seus objetos instanciados.

Exemplo:

```
#include "protheus.ch"
CLASS Pessoa()

DATA cNome
DATA nIdade
```

### **Prototipação dos métodos**

A prototipação dos métodos é uma regra utilizada pelas linguagens orientadas a objetos, através da qual são especificadas as operações que podem ser realizadas pelo objeto, diferenciando os métodos de outras funções internas de uso da classe, e para especificar quais são os métodos construtores.

Em linguagens tipadas, na prototipação dos métodos é necessário definir quais são os parâmetros recebidos e seus respectivos tipos, além de definir o tipo do retorno que será fornecido. Em ADVPL é necessário apenas descrever a chamada do método e caso necessário se o mesmo é um construtor.

Exemplo:

```
#include "protheus.ch"
CLASS Pessoa()

DATA cNome
DATA nIdade

METHOD Create() CONSTRUCTOR
METHOD SetNome()
METHOD SetIdade()

ENDCLASS
```

## **2.3. Implementação dos métodos de uma classe em ADVPL**

### **Método Construtor**

O método construtor possui a característica de retornar um objeto com o tipo da classe da qual o mesmo foi instanciado. Por esta razão diz-se que o tipo do objeto instanciado é a classe daquele objeto.

Para produzir este efeito, o método construtor utiliza a palavra reservada “SELF”, a qual é utilizada pela linguagem ADVPL para referência a própria classe daquele objeto.

Exemplo:

```
#include "protheus.ch"
CLASS Pessoa()

DATA cNome
DATA nIdade

METHOD Create() CONSTRUCTOR
METHOD SetNome()
METHOD SetIdade()

ENDCLASS

METHOD Create(cNome, nIdade) CLASS Pessoa

::cNome := cNome
::nIdade := nIdade

Return SELF
```

## **Manipulação de atributos**

---

Os atributos definidos para uma classe com a utilização da palavra reservada “DATA” em sua declaração podem ser manipulados por seus métodos utilizando o operador “::”.

A utilização deste operador permite ao interpretador ADVPL diferenciar variáveis comuns criadas pelas funções e métodos que utilizam este objeto dos atributos propriamente ditos.

Exemplo:

```
#include "protheus.ch"
CLASS Pessoa()

DATA cNome
DATA nIdade

METHOD Create() CONSTRUCTOR
METHOD SetNome()
METHOD SetIdade()
ENDCLASS

METHOD Create(cNome, nIdade) CLASS Pessoa

::cNome := cNome
::nIdade := nIdade

Return SELF
```

## **Utilização de funções em uma classe de objetos**

---

Conforme mencionado anteriormente, a utilização da palavra reservada “METHOD” permite ao interpretador ADVPL diferenciar os métodos que podem ser utilizados através da referência do objeto de funções internas descritas internamente na classe.

Isto permite a utilização de funções tradicionais da linguagem ADVPL, como as Static Functions() as quais serão visíveis apenas a classe, e não poderão ser referenciadas diretamente pelo objeto.

Exemplo - parte 01: Função CadPessoa (usuária da classe Pessoa)

```
#include "protheus.ch"

USER FUNCTION CadPessoa()

Local oPessoa
Local cNome := ""
Local dNascimento:= CTOD("")
Local aDados := {}

aDados := GetDados()
oPessoa := Pessoa():Create(cNome, dNascimento)

Return
```

```

Exemplo - parte 02: Classe Pessoa

#include "protheus.ch"
CLASS Pessoa()

DATA cNome
DATA nIdade
DATA dNascimento

METHOD Create() CONSTRUCTOR
METHOD SetNome()
METHOD SetIdade()

ENDCLASS

METHOD Create(cNome, dNascimento) CLASS Pessoa
::cNome := cNome
::dNascimento := dNascimento
::nIdade := CalcIdade(dNascimento)
Return SELF

STATIC FUNCTION CalcIdade(dNascimento)
Local nIdade
nIdade := dDataBase - dNascimento
RETURN nIdade

```

### **Herança entre classes**

Seguindo o princípio da orientação a objetos, a linguagem ADVPL permite que uma classe receba por herança os métodos e atributos definidos em uma outra classe, a qual tornasse a superclasse desta instância.

Para utilizar este recurso deve ser utilizada a palavra reservada “FROM” na declaração da classe, especificando a superclasse que será referenciada.

Em ADVPL o exemplo prático desta situação é a superclasse TSrvObject, a qual é utilizada pela maioria das classes e componentes da interface visual, como demonstrado no módulo 06.

```

Exemplo - parte 01: Declaração da classe Pessoa

#include "protheus.ch"
CLASS Pessoa()

DATA cNome
DATA nIdade
DATA dNascimento

METHOD Create() CONSTRUCTOR
METHOD SetNome()
METHOD SetIdade()

ENDCLASS

```

```

Exemplo - parte 02: Declaração da classe Aluno

#include "protheus.ch"
CLASS Aluno() FROM Pessoa

DATA nID
DATA aCursos

METHOD Create() CONSTRUCTOR
METHOD Inscrever()
METHOD Avaliar()
METHOD GetNota()
METHOD GetStatus()

ENDCLASS

// Os objetos da classe Aluno, possuem todos os métodos e atributos da classe
Pessoa, além
// dos métodos e atributos declarados na própria classe.

```

### **Construtor para classes com herança**

Quanto é utilizado o recurso de herança entre classes, o construtor da classe instanciada deve receber um tratamento adicional, para que o objeto instanciado seja criado com os atributos e métodos definidos na superclasse.

Nestes casos, logo após a definição do método construtor da classe, deverá ser executado o método construtor da superclasse.

```

Exemplo - parte 03: Método Construtor da classe Aluno

METHOD Create(cNome,dNascimento,nID)
:Create(cNome,dNascimento) // Chamada do método construtor da classe Pessoa.

::nID := ID

Return SELF

```

### 3. Regras adicionais da linguagem ADVPL

#### 3.1. Palavras reservadas

AADD	DTOS	INKEY	REPLICATE	VAL
ABS	ELSE	INT	RLOCK	VALTYPE
ASC	ELSEIF	LASTREC	ROUND	WHILE
AT	EMPTY	LEN	ROW	WORD
BOF	ENDCASE	LOCK	RTRIM	YEAR
BREAK	ENDDO	LOG	SECONDS	CDOW
ENDIF	LOWER	SELECT	CHR	EOF
LTRIM	SETPOS	CMONTH	EXP	MAX
SPACE	COL	FCOUNT	MIN	SQRT
CTOD	FIELDNAME	MONTH	STR	DATE
FILE	PCOL	SUBSTR	DAY	FLOCK
PCOUNT	TIME	DELETED	FOUND	PROCEDURE
TRANSFORM	DEVPOS	FUNCTION	PROW	TRIM
DOW	IF	RECCOUNT	TYPE	DTOC
IIF	RECNO	UPPER	TRY	AS
CATCH	THROW			



Importante

- Palavras reservadas não podem ser utilizadas para variáveis, procedimentos ou funções;
- Funções reservadas são pertencentes ao compilador e não podem ser redefinidas por uma aplicação;
- Todos os identificadores que começarem com dois ou mais caracteres “\_” são utilizados como identificadores internos e são reservados.
- Identificadores de escopo PRIVATE ou PUBLIC utilizados em aplicações específicas desenvolvida por ou para clientes devem ter sua identificação iniciada por um caractere “\_”.

### **3.2. Pictures de formatação disponíveis**

Com base na documentação disponível no DEM – Documentação Eletrônica Microsiga, a linguagem ADVPL e a aplicação ERP Protheus admitem as seguintes pictures:

#### **Dicionário de Dados (SX3) e GET**

<b>Funções</b>	
<b>Conteúdo</b>	<b>Funcionalidade</b>
<b>A</b>	Permite apenas caracteres alfabéticos.
<b>C</b>	Exibe CR depois de números positivos.
<b>E</b>	Exibe numérico com o ponto e vírgula invertidos (formato Europeu).
<b>R</b>	Insere caracteres diferentes dos caracteres de template na exibição, mas não os insere na variável do GET.
<b>S&lt;n&gt;</b>	Permite rolamento horizontal do texto dentro do GET, <n> é um número inteiro que identifica o tamanho da região.
<b>X</b>	Exibe DB depois de números negativos.
<b>Z</b>	Exibe zeros como brancos.
<b>(</b>	Exibe números negativos entre parênteses com os espaços em branco iniciais.
<b>)</b>	Exibe números negativos entre parênteses sem os espaços em branco iniciais.
<b>!</b>	Converte caracteres alfabéticos para maiúsculo.

<b>Templates</b>	
<b>Conteúdo</b>	<b>Funcionalidade</b>
<b>X</b>	Permite qualquer caractere.
<b>9</b>	Permite apenas dígitos para qualquer tipo de dado, incluindo o sinal para numéricos.
<b>#</b>	Permite dígitos, sinais e espaços em branco para qualquer tipo de dado.
<b>!</b>	Converte caracteres alfabéticos para maiúsculo.
<b>*</b>	Exibe um asterisco no lugar dos espaços em branco iniciais em números.
<b>.</b>	Exibe o ponto decimal.
<b>,</b>	Exibe a posição do milhar.

Exemplo 01 – Picture campo numérico

CT2\_VALOR – Numérico – 17,2  
Picture: @E 99,999,999,999,999.99

Exemplo 02 – Picture campo texto, com digitação apenas em caixa alta

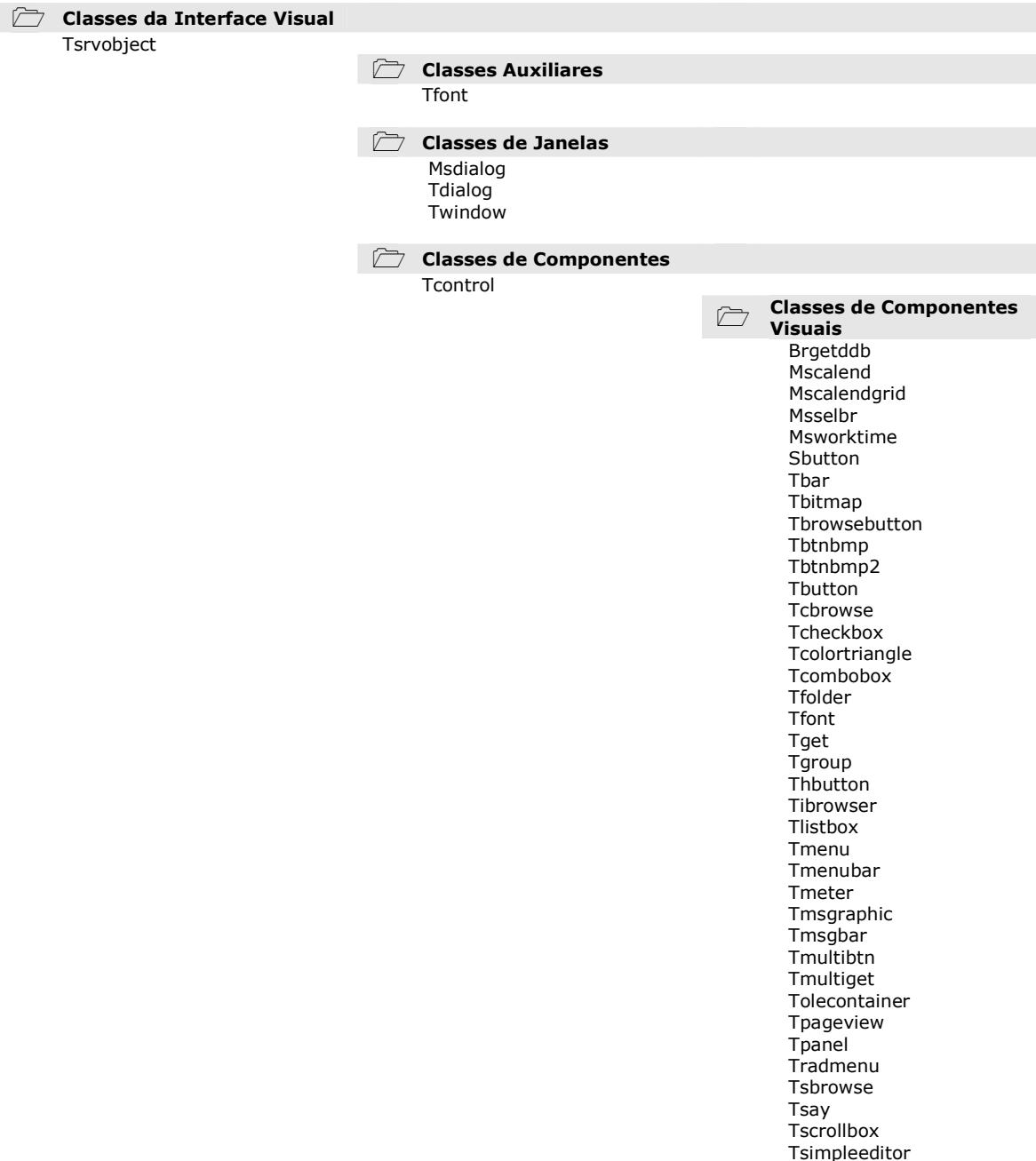
A1\_NOME – Caracter – 40  
Picture: @!

## MÓDULO 07: ADVPL Orientado à objetos II

Neste módulo serão tratados os componentes e objetos da interface visual da linguagem ADVPL, permitindo o desenvolvimento de aplicações com interfaces gráficas com sintaxe orientada a objetos.

### 4. Componentes da interface visual do ADVPL

A linguagem ADVPL possui diversos componentes visuais e auxiliares, os quais podem ser representados utilizando a estrutura abaixo:



	<b>Classes de Componentes Visuais</b>
	Tsplitter
	Ttabs
	Ttoolbox
	Twbrowse
	Vcbrowse

### Atributos comuns as classes de componentes visuais

<b>cCaption</b>	Título ou conteúdo do objeto.
<b>cF3</b>	Consulta F3 ao qual o objeto está vinculado, quando o mesmo é do tipo get.
<b>cMsg</b>	Mensagem exibida na barra de status da janela principal quando o objeto ganha foco.
<b>cName</b>	Nome do objeto
<b>cTooltip</b>	Mensagem exibida quando objeto exibe seu tooltip.
<b>cVariable</b>	Variável que receberá o conteúdo selecionado ou digitado no objeto, quando o mesmo for do tipo get ou similar.
<b>IPassword</b>	Indica se o texto digitado no objeto será exibido no formato de password, quando o mesmo é do tipo get.
<b>IReadOnly</b>	Flag que indica se o objeto pode ou não ser editado.
<b>IShowHint</b>	Flag que ativa .T. ou desativa .F. a exibição do tooltip do objeto.
<b>IVisibleControl</b>	Se .T. o objeto é visível, se .F. o objeto é invisível.
<b>nHeight</b>	Altura em pixels.
<b>nLeft</b>	Coordenada horizontal em pixels.
<b>nTop</b>	Coordenada vertical em pixels.
<b>nWidth</b>	Largura em pixels.
<b>Picture</b>	Picture de digitação do objeto, quando o mesmo é do tipo get.

### Eventos comuns as classes de componentes visuais

<b>bAction</b>	Bloco de código executado quando o objeto é utilizado, quando o mesmo é do tipo botão ou similar.
<b>bChange</b>	Bloco de código executado quando ocorre uma mudança na seleção de itens que compõe o objeto, como por exemplo em listas ou grids.
<b>bF3</b>	Bloco de código executado quando utilizado o recurso de consulta F3 vinculado ao objeto.
<b>bGotFocus</b>	Executado quando objeto ganha foco.
<b>bIClicked</b>	Executado quando acionado click do botão esquerdo do mouse sobre o objeto.
<b>bIDbIClick</b>	Executado quando acionado duplo click do botão esquerdo do mouse sobre o objeto.
<b>bLostFocus</b>	Executado quando objeto perde foco.
<b>brClicked</b>	Executado quando acionado click do botão direito do mouse sobre o objeto.
<b>bValid</b>	Executado quando o conteúdo do objeto é modificado e deverá ser validado. Deve retornar .T. se o conteúdo é válido e .F. se conteúdo inválido.
<b>bWhen</b>	Executado quando há movimentação de foco na janela. Se retornar .T. o objeto continua habilitado, se retornar .F. o

	objeto será desabilitado.
--	---------------------------

## Classes da interface visual

### TSRVOBJECT()

- Descrição:** Classe abstrata inicial de todas as classes de interface do ADVPL. Não deve ser instanciada diretamente.

- Propriedades:**

<b>nLeft</b>	Coordenada horizontal em pixels.
<b>nTop</b>	Coordenada vertical em pixels.
<b>nWidth</b>	Largura em pixels.
<b>nHeight</b>	Altura em pixels.
<b>cCaption</b>	Título ou conteúdo do objeto.
<b>cTooltip</b>	Mensagem exibida quando objeto exibe seu tooltip.
<b>IShowHint</b>	Flag que ativa .T. ou desativa .F. a exibição do tooltip do objeto.
<b>cMsg</b>	Mensagem exibida na barra de status da janela principal quando o objeto ganha foco.
<b>nClrText</b>	Cor do texto do objeto.
<b>nClrPane</b>	Cor do fundo do objeto.
<b>bWhen</b>	Executado quando há movimentação de foco na janela. Se retornar .T. o objeto continua habilitado, se retornar .F. o objeto será desabilitado.
<b>bValid</b>	Executado quando o conteúdo do objeto é modificado e deverá ser validado. Deve retornar .T. se o conteúdo é válido e .F. se conteúdo inválido.
<b>bIClicked</b>	Executado quando acionado click do botão esquerdo do mouse sobre o objeto.
<b>brClicked</b>	Executado quando acionado click do botão direito do mouse sobre o objeto.
<b>bIDbIClick</b>	Executado quando acionado duplo click do botão esquerdo do mouse sobre o objeto.
<b>oWnd</b>	Janela onde o objeto foi criado.
<b>IVisible</b>	Se .T. o objeto é visível, se .F. o objeto é invisível.
<b>Cargo</b>	Conteúdo associado ao objeto.
<b>bLostFocus</b>	Executado quando objeto perde foco.
<b>bGotFocus</b>	Executado quando objeto ganha foco.

- Construtor: Não pode ser instanciada.**

- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

**Métodos auxiliares:**

**SetFocus**

---

- Descrição:** Força mudança do foco de entrada de dados para o controle.
- Sintaxe:** SetFocus( )
- Parâmetros:**

**Nenhum** -

- Retorno:**

**Nil**

**Hide**

---

- Descrição:** Torna objeto invisível.
- Sintaxe:** Hide( )
- Parâmetros:**

**Nenhum** -

- Retorno:**

**Nil**

**Show**

---

- Descrição:** Torna objeto visível.
- Sintaxe:** Show( )
- Parâmetros:**

**Nenhum** -

- Retorno:**

**Nil**

**Enable**

---

- Descrição:** Habilita o objeto.
- Sintaxe:** Enable( )
- Parâmetros:**

**Nenhum** -

- Retorno:**

**Nil**

## **Disable**

---

- Descrição:** Desabilita o objeto.
- Sintaxe:** Disable( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nil</b>	
------------	--

## **Refresh**

---

- Descrição:** Força atualização (sincronia) de propriedades entre o programa e o Protheus Remote.
- Sintaxe:** Refresh( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nil</b>	
------------	--

## **Exemplo:**

---

CLASS TCONTROL FROM TSRVOBJECT
--------------------------------

## Classes auxiliares

### TFONT()

- Descrição:** Classe de objetos que define a fonte do texto utilizado nos controles visuais.

- Propriedades:**

-	Herdadas as classes superiores
---	--------------------------------

- Construtor:** New([acName], [nPar2], [anHeight], [IPar4], [alBold], [nPar6], [IPar7], [nPar8], [alItalic], [alUnderline])

- Parâmetros:**

<b>acName</b>	Caractere, opcional. Nome da fonte, o padrão é "Arial".
<b>nPar2</b>	Reservado.
<b>anHeight</b>	Numérico, opcional. Tamanho da fonte. O padrão é -11.
<b>IPar4</b>	Reservado.
<b>alBold</b>	Lógico, opcional. Se .T. o estilo da fonte será negrito.
<b>nPar6</b>	Reservado.
<b>IPar7</b>	Reservado.
<b>nPar8</b>	Reservado.
<b>alItalic</b>	Lógico, opcional. Se .T. o estilo da fonte será itálico.
<b>alUnderline</b>	Lógico, opcional. Se .T. o estilo da fonte será sublinhado.

- Aparência:**



### Exemplo:

```
#include 'protheus.ch'
User Function Teste()
Local oDlg, oSay
DEFINE MSDIALOG oDlg FROM 0,0 TO 200,200 TITLE 'My dialog' PIXEL
// Cria font para uso
oFont:= TFont():New('Courier New',,-14,.T.)
// Apresenta o tSay com a fonte Courier New
oSay := TSay():New( 10, 10, {|| 'Mensagem'},oDlg,, oFont,,, .T.,
CLR_WHITE,CLR_RED )
oSay:lTransparent:= .F.
ACTIVATE MSDIALOG oDlg CENTERED
Return
```

## Classes de janelas

### **MSDIALOG()**

- Descrição:** Classe de objetos que deve ser utilizada como padrão de janela para entrada de dados. MSDialog é um tipo de janela diálogo modal, isto é, não permite que outra janela ativa receba dados enquanto esta estiver ativa.

- Propriedades:**

-	Herdadas as classes superiores
---	--------------------------------

- Construtor:** New([anTop], [anLeft], [anBottom], [anRight], [acCaption], [cPar6], [nPar7], [IPar8], [nPar9], [anClrText], [anClrBack], [oPar12], [aoWnd], [alPixel], [oPar15], [oPar16], [IPar17])

- Parâmetros:**

<b>anTop</b>	Numérico, opcional. Coordenada vertical superior em pixels ou caracteres.
<b>anLeft</b>	Numérico, opcional. Coordenada horizontal esquerda em pixels ou caracteres.
<b>anBottom</b>	Numérico, opcional. Coordenada vertical inferior em pixels ou caracteres.
<b>anRight</b>	Numérico, opcional. Coordenada horizontal direita em pixels ou caracteres.
<b>acCaption</b>	Caractere, opcional. Título da janela.
<b>cPar6</b>	Reservado.
<b>nPar7</b>	Reservado.
<b>IPar8</b>	Reservado.
<b>nPar9</b>	Reservado.
<b>anClrText</b>	Numérico,opcional. Cor do texto.
<b>anClrBack</b>	Numérico,opcional. Cor de fundo.
<b>oPar12</b>	Reservado.
<b>aoWnd</b>	Objeto, opcional. Janela mãe da janela a ser criada, padrão é a janela principal do programa.
<b>alPixel</b>	Lógico, opcional. Se .T. considera as coordenadas passadas em pixels, se .F. considera caracteres.
<b>oPar15</b>	Reservado.
<b>oPar16</b>	Reservado.
<b>nPar17</b>	Reservado.

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
// cria diálogo
Local oDlg := MSDialog():New(10,10,300,300,'Meu
dialogo',,,,CLR_BLACK,CLR_WHITE,,,T.)

// ativa diálogo centralizado
oDlg:Activate(,,,T.,{||msgstop('validou!'),.T.},,{||msgstop('iniciando...
}) )
Return
```

**TDIALOG()**

- Descrição:** Classe de objetos do tipo diálogo de entrada de dados, sendo seu uso reservado. Recomenda-se utilizar a classe MSDialog que é herdada desta classe.

**Propriedades:**

-	Herdadas as classes superiores
---	--------------------------------

- Construtor:** New([anTop], [anLeft], [anBottom], [anRight], [acCaption], [cPar6], [nPar7], [IPar8], [nPar9], [anClrText], [anClrBack], [oPar12], [aoWnd], [alPixel], [oPar15], [oPar16], [nPar17], [anWidth], [anHeight])

**Parâmetros:**

<b>anTop</b>	Numérico, opcional. Coordenada vertical superior em pixels ou caracteres.
<b>anLeft</b>	Numérico, opcional. Coordenada horizontal esquerda em pixels ou caracteres.
<b>anBottom</b>	Numérico, opcional. Coordenada vertical inferior em pixels ou caracteres.
<b>anRight</b>	Numérico, opcional. Coordenada horizontal direita em pixels

	ou caracteres.
<b>acCaption</b>	Caractere, opcional. Título da janela.
<b>cPar6</b>	Reservado.
<b>nPar7</b>	Reservado.
<b>IPar8</b>	Reservado.
<b>nPar9</b>	Reservado.
<b>anClrText</b>	Numérico,opcional. Cor do texto.
<b>anClrBack</b>	Numérico,opcional. Cor de fundo.
<b>oPar12</b>	Reservado.
<b>aoWnd</b>	Objeto, opcional. Janela mãe da janela a ser criada, padrão é a janela principal do programa.
<b>alPixel</b>	Lógico, opcional. Se .T. considera as coordenadas passadas em pixels, se .F. considera caracteres.
<b>oPar15</b>	Reservado.
<b>oPar16</b>	Reservado.
<b>nPar17</b>	Reservado.
<b>anWidth</b>	Numérico, opcional. Largura da janela em pixels.
<b>anHeight</b>	Numérico, opcional. Altura da janela em pixels.



### Métodos auxiliares:

#### Activate

---

- Descrição:** Ativa (exibe) o diálogo. Chamar somente uma vez este método.

- Sintaxe:** Activate([bPar1], [bPar2], [bPar3], [alCentered], [abValid], [IPar6], [abInit], [bPar8], [bPar9] )

- Parâmetros:**

<b>bPar1</b>	Reservado.
<b>bPar2</b>	Reservado.
<b>bPar3</b>	Reservado.
<b>alCentered</b>	Lógico, opcional. Se .T. exibe a janela centralizada, .F. é padrão.
<b>abValid</b>	Bloco de código, opcional. Deve retornar .T. se conteúdo do diálogo é válido, se retornar .F. o diálogo não fechará quando solicitada de encerrar.
<b>IPar6</b>	Reservado.
<b>abInit</b>	Bloco de código, opcional. Executado quando o diálogo inicia exibição.
<b>bPar8</b>	Reservado.
<b>bPar9</b>	Reservado.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **End**

---

- Descrição:** Encerra (fecha) o diálogo.
- Sintaxe:** End( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:** Lógico .T. se o diálogo foi encerrado.

<b>Lógico</b>	Indica se o diálogo foi encerrado.
---------------	------------------------------------

**Aparência:**



**Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
  // cria diálogo
  oDlg := MSDialog():New(10,10,300,300,'Meu
dialogo',,,,CLR_BLACK,CLR_WHITE,,,T.)
  // ativa diálogo centralizado
  oDlg:Activate(,,,T.,{|msgstop('validou!'),.T.},,{||msgstop('iniciando...
.')}) )

/* os comandos abaixo proporcionam o mesmo resultado
// cria diálogo
DEFINE DIALOG oDlg TITLE 'Meu dialogo' FROM 10,10 TO 300,300 COLOR
CLR_BLACK,CLR_WHITE PIXEL
// ativa diálogo centralizado
ACTIVATE DIALOG oDlg CENTER ON INIT (msgstop('iniciando...')) VALID
(msgstop('validou!'),.T.)
*/
Return Nil
```

## **TWINDOW()**

---

- Descrição:** Classe de objetos do tipo diálogo principal de programa. Deverá existir apenas uma instância deste objeto na execução do programa.

**Propriedades:**

-	Herdadas das classes superiores
---	---------------------------------

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

**Parâmetros:**

<b>nTop</b>	Numérico, opcional. Coordenada vertical superior em pixels ou caracteres.
<b>nLeft</b>	Numérico, opcional. Coordenada horizontal esquerda em pixels ou caracteres.
<b>nBottom</b>	Numérico, opcional. Coordenada vertical inferior em pixels ou caracteres.
<b>nRight</b>	Numérico, opcional. Coordenada horizontal inferior em pixels ou caracteres.
<b>cTitle</b>	Caractere, opcional. Título da janela.
<b>nPar6</b>	Reservado.
<b>oPar7</b>	Reservado.
<b>oPar8</b>	Reservado.
<b>oPar9</b>	Reservado.
<b>oParent</b>	Objeto, opcional. Janela mãe da janela corrente.
<b>IPar11</b>	Reservado.
<b>IPar12</b>	Reservado.
<b>nClrFore</b>	Numérico, opcional. Cor de fundo da janela.
<b>nClrText</b>	Numérico, opcional. Cor do texto da janela.
<b>oPar15</b>	Reservado.
<b>cPar16</b>	Reservado.
<b>IPar17</b>	Reservado.
<b>IPar18</b>	Reservado.
<b>IPar19</b>	Reservado.
<b>IPar20</b>	Reservado.
<b>IPixel</b>	Lógico, opcional. Se .T. (padrão) considera coordenadas passadas em pixels, se .F. considera caracteres.

**Métodos auxiliares:**

**Activate**

---

- Descrição:** 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]).
- Parâmetros:**

<b>acShow</b>	Caracter, opcional. "ICONIZED" para janela iconizada ou "MAXIMIZED" para janela maximizada.
<b>bPar2</b>	Reservado.
<b>bPar3</b>	Reservado.
<b>bPar4</b>	Reservado.
<b>bPar5</b>	Reservado.
<b>bPar6</b>	Reservado.
<b>abInit</b>	Bloco de código. Executado quando janela está sendo exibida.
<b>bPar8</b>	Reservado.
<b>bPar9</b>	Reservado.
<b>bPar10</b>	Reservado.
<b>bPar11</b>	Reservado.
<b>bPar12</b>	Reservado.
<b>bPar13</b>	Reservado.
<b>bPar14</b>	Reservado.
<b>bPar15</b>	Reservado.
<b>abValid</b>	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.
<b>bPar17</b>	Reservado.
<b>bPar18</b>	Reservado.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**End**

---

- Descrição:** Solicita encerramento da janela.
- Sintaxe:** End( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:** Lógico .T. se o diálogo foi encerrado.

<b>Lógico</b>	Indica se a janela foi encerrada.
---------------	-----------------------------------

## Center

---

- Descrição:** Centraliza a janela.
- Sintaxe:** Center( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:** Lógico .T. se o diálogo foi encerrado.

<b>Nenhum</b>	-
---------------	---

**Aparência:**



**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
.
```

## Classes de componentes

### TCONTROL()

- Descrição:** Classe abstrata comum entre todos os componentes visuais editáveis. Não deve ser instanciada diretamente.

- Propriedades:**

<b>Align</b>	Numérico. Alinhamento do controle no espaço disponibilizado pelo seu objeto parente. 0 = Nenhum (padrão), 1= no topo, 2 = no rodapé, 3= a esquerda, 4 = a direita e 5 = em todo o parente.
<b>IModified</b>	Lógico. Se .T. indica que o conteúdo da variável associada ao controle foi modificado.
<b>IReadOnly</b>	Lógico. Se .T. o conteúdo da variável associada ao controle permanecerá apenas para leitura.
<b>hParent</b>	Numérico. Handle (identificador) do objeto sobre o qual o controle foi criado.
<b>bChange</b>	Bloco de código. Executado quando o estado ou conteúdo do controle é modificado pela ação sobre o controle.

- Construtor: Não pode ser instanciada.**

- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Métodos auxiliares:**

#### SetFocus

- Descrição:** Força mudança do foco de entrada de dados para o controle.

- Sintaxe:** SetFocus( )

- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nil</b>
------------

#### Exemplo:

```
CLASS TSay FROM Tcontrol
```

## Classes de componentes visuais

### BRGETDDB()

- Descrição:** Classe de objetos visuais do tipo Grid.

- Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>nAt</b>	Linha atualmente selecionada / posicionada no objeto
<b>nLen</b>	Número total de linhas do objeto

- Construtor:** New([nRow], [nCol], [nWidth], [nHeighth], [bFields], [aHeaders], [aColSizes], [oDlg], [cField], [uValue1], [uValue2], [uChange], [uLDbClick], [uRCClick], [oFont], [oCursor], [nClrFore], [nClrBack], [cMsg], [lUpdate], [cAlias], [IPixel], [bWhen], [IDesign], [bValid], [aAlter], [oMother])

- Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical
<b>nCol</b>	Numérico, opcional. Coordenada horizontal
<b>nWidth</b>	Numérico, opcional. Largura do objeto
<b>nHeight</b>	Numérico, opcional. Altura do objeto
<b>bFields</b>	Bloco de código, Lista de Campos
<b>aHeaders</b>	Vetor, Descrição dos campos para no cabeçalho
<b>aColSizes</b>	Vetor, Largura das colunas
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
<b>cField</b>	Caracter, opcional. Campos necessários ao filtro
<b>uValue1</b>	Indefinido, opcional. Início do intervalo para o filtro
<b>uValue2</b>	Indefinido, opcional. Fim do intervalo para o filtro
<b>bChange</b>	Bloco de código, opcional. Executado quando o item selecionado é alterado.
<b>bLDbClick</b>	Bloco de código, opcional. Executado quando acionado duplo click do botão esquerdo do mouse sobre o controle.
<b>bRCClick</b>	Não utilizado
<b>oFont</b>	Objeto, opcional. Fonte
<b>oCursor</b>	Objeto, opcional. Tipo do Cursor
<b>nClrFore</b>	Numérico, opcional. Cor do texto da janela.
<b>nClrBack</b>	Numérico, opcional. Cor de fundo da janela.
<b>cMsg</b>	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
<b>lUpdate</b>	Não utilizado
<b>cAlias</b>	Caracter, opcional se objeto utilizado com Vetor, obrigatório se utilizado com Tabela
<b>IPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IDesign</b>	Não Utilizado

<b>bValid</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.
<b>aAlter</b>	Não Utilizado
<b>oMother</b>	Não Utilizado

**Métodos auxiliares:**

#### **GoUp**

---

- Descrição:** Salta uma linha para cima.
- Sintaxe:** GoUp( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

#### **GoDown**

---

- Descrição:** Salta uma linha para baixo.
- Sintaxe:** GoDown( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

#### **GoTop**

---

- Descrição:** Salta para primeira linha.
- Sintaxe:** GoTop( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

#### **GoBottom**

---

- Descrição:** Salta para ultima linha.
- Sintaxe:** GoBottom( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

**Nil**

### RowCount

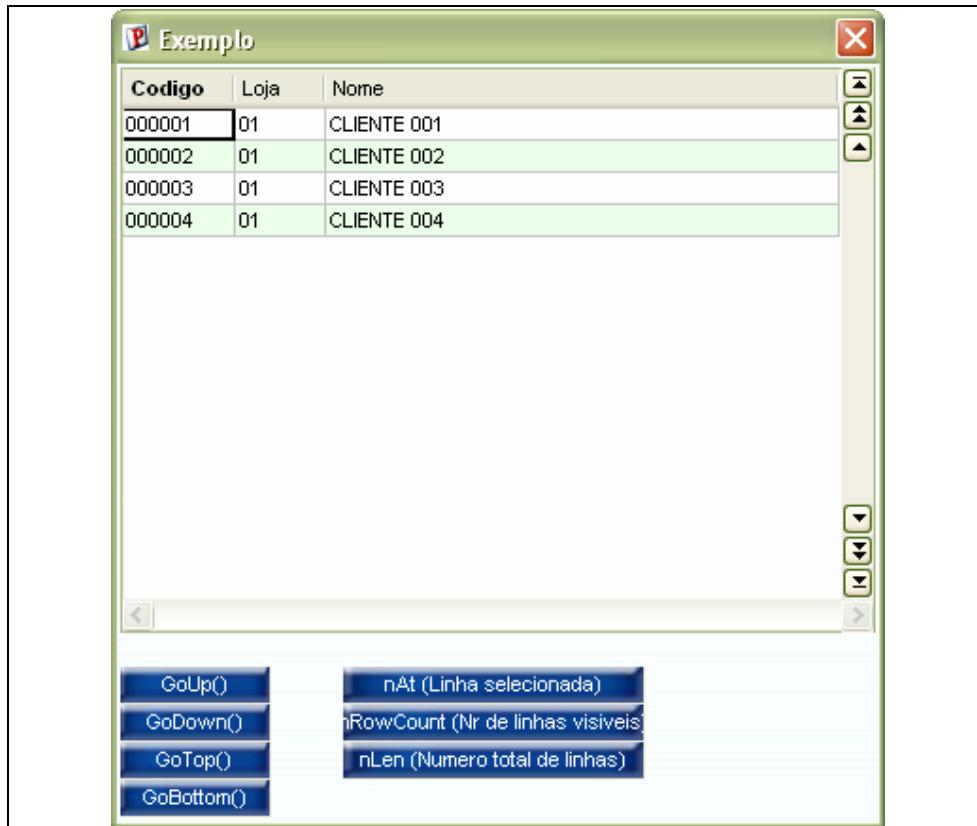
- Descrição:** Retorna numero de linhas visiveis.
- Sintaxe:** RowCount( )
- Parâmetros:**

Nenhum

- Retorno:**

**Nil**

**Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 302,402 PIXEL TITLE 'Exemplo'

        DbSelectArea('SA1')
        oBrowse := BrGetDDB():New(
1,1,200,150,,,.oDlg,,,.F.,,'SA1',.T.,,.F.,,, )
        oBrowse:addColumn(TCColumn():New('Codigo',{||SA1->A1_COD
},,'LEFT',,.F.,.F.,,.F.,))
        oBrowse:addColumn(TCColumn():New('Loja' ,{||SA1-
>A1_LOJA},,'LEFT',,.F.,.F.,,.F.,))
        oBrowse:addColumn(TCColumn():New('Nome' ,{||SA1-
>A1_NOME},,'LEFT',,.F.,.F.,,.F.,))

        // Principais comandos
        TButton():New(160,001,'GoUp()',oDlg,{|| oBrowse:GoUp()
},40,10,,,.T.)
        TButton():New(170,001,'GoDown()',oDlg,{|| oBrowse:GoDown()
},40,10,,,.T.)
        TButton():New(180,001,'GoTop()',oDlg,{|| oBrowse:GoTop()
},40,10,,,.T.)
        TButton():New(190,001,'GoBottom()' , oDlg,{|| oBrowse:GoBottom()
},40,10,,,.T.)
        TButton():New(160,060,'nAt (Linha selecionada)',oDlg,;
            {|| Alert(oBrowse:nAt)},80,10,,,.T.)
        TButton():New(170,060,'nRowCount (Nr de linhas visiveis)',oDlg,;
            {|| Alert(oBrowse:nRowCount()) },80,10,,,.T.)
        TButton():New(180,060, 'nLen (Número total de linhas)' , oDlg,;
            {|| Alert(oBrowse:nLen) },80,10,,,.T.)

        ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

---

## **MSCALEND()**

- Descrição:** Classe de objetos visuais do tipo Calendário.

- Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>bChange</b>	Bloco de código executado na mudança de seleção de um dia.
<b>bChangeMes</b>	Bloco de código executado na mudança de seleção de um mês.
<b>dDiaAtu</b>	Dia atualmente selecionado

- Construtor:** New([nRow], [nCol], [oDlg], [ICanMultSel])

**Parâmetros:**

nRow	Numérico, opcional. Coordenada vertical
nCol	Numérico, opcional. Coordenada horizontal
oDlg	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
ICanMultSel	Logico, opcional. Permite seleção de multiplos dias

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
user function Calend_Ex()
    DEFINE MSDIALOG oDlg FROM 0,0 TO 1000,1000 PIXEL TITLE 'Exemplo de
MsCalend'
    // Cria objeto
    oCalend:=MsCalend():New(01,01,oDlg,.T.)
    // Code-Block para mudança de Dia
    oCalend:bChange := {|| Alert('Dia Selecionado: ' +
dtoc(oCalend:dDiaAtu)) }
    // Code-Block para mudança de mes
    oCalend:bChangeMes := {|| alert('Mes alterado') }
    ACTIVATE MSDIALOG oDlg CENTERED
Return
```

## **MSCALENDGRID()**

---

- Descrição:** Classe de objetos visuais do tipo Grade de Períodos.

**Propriedades:**

<b>bAction</b>	Bloco de código. Executado quando o botão esquerdo do mouse é pressionado.
<b>bRClick</b>	Bloco de código. Executado quando o botão direito do mouse é pressionado.

- Construtor:** New([aoWnd], [anRow], [anCol], [anHeight], [anWidth], [dDateIni], [nResolution], [abWhen], [abAction], [nDefColor], [bRClick], [IFillAll])

**Parâmetros:**

<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>anHeight</b>	Numérico, opcional. Altura do botão em pixels.
<b>anWidth</b>	Numérico, opcional. Largura do botão em pixels.
<b>dDateIni</b>	Data, Data inicial do Calendário
<b>nResolution</b>	Numérico, Resolução a ser aplicada na grid do Calendário
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>abAction</b>	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.
<b>nDefColor</b>	Numérico, opcional. Cor do Fundo da Grid.
<b>bRClick</b>	Bloco de código, opcional. Executado quando acionado o botão direito do mouse sobre o controle.
<b>IFillAll</b>	Lógico, opcional. Preenche todo o periodo

**Métodos auxiliares:**

---

**Add**

- Descrição:** Adiciona periodo na Grid.

- Sintaxe:** Add(cCaption, nLin, nInitial, nFinal, nColor, cDescri )

**Parâmetros:**

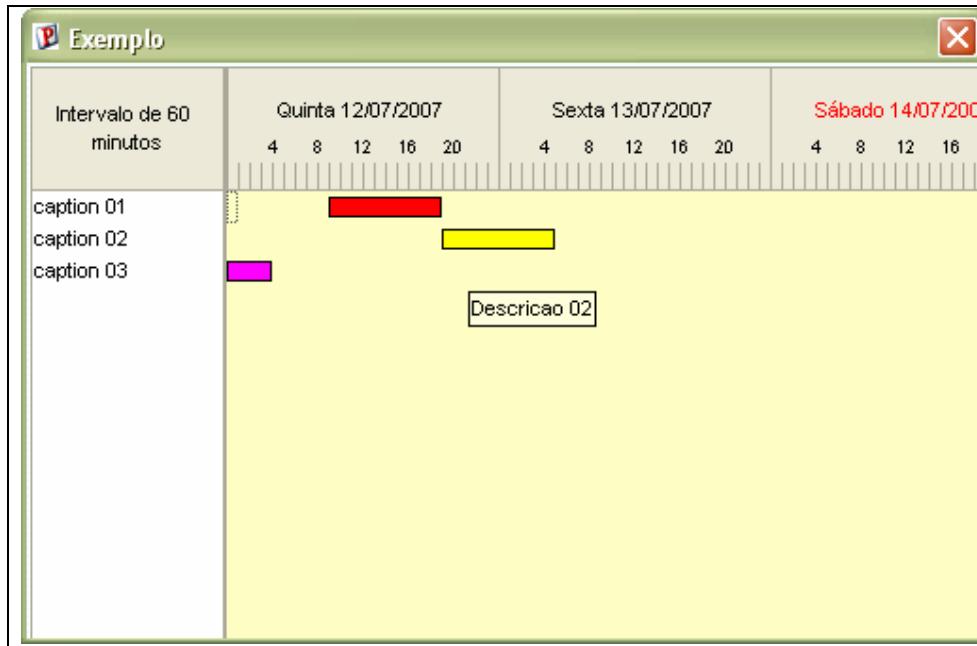
<b>cCaption</b>	Caracter. Descrição a ser inserida a esquerda da Grid
<b>nLin</b>	Numerico. Numero da linha a ser inserido o item
<b>nInitial</b>	Numerico. Dia inicial
<b>nFinal</b>	Numerico. Dia Final

<b>nColor</b>	Numerico. Cor que destaca o item
<b>cDescri</b>	Caracter, Opcional. Descrição do tipo tooltip(hint) do item

**Retorno:**

Nil	
-----	--

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    nResolution := 1
    oMsCalendGrid := MsCalendGrid():New( oDlg, 01, 01, 500,300,;
                                         date(), nResolution, ,{|x,y| Alert(x)},;
                                         RGB(255,255,196), {|x,y|Alert(x,y)}, .T.
)
    oMsCalendGrid:Add('caption 01', 1, 10, 20, RGB(255,0,0,0),
'Descricao 01')
    oMsCalendGrid:Add('caption 02', 2, 20, 30, RGB(255,255,0),
'Descricao 02')
    oMsCalendGrid:Add('caption 03', 3, 01, 05, RGB(255,0,255),
'Descricao 03')
    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **MSSELBR()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Grid

- Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>nAt</b>	Linha atualmente selecionada / posicionada no objeto
<b>nLen</b>	Número total de linhas do objeto

- Construtor:** New([nRow], [nCol], [nWidth], [nHeighth], [bFields], [aHeaders], [aColSizes], [oDlg], [cField], [uValue1], [uValue2], [uChange], [uLDbClick], [uRClick], [oFont], [oCursor], [nClrFore], [nClrBack], [cMsg], [IUpdate], [cAlias], [IPixel], [bWhen], [IDesign], [bValid])

- Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical
<b>nCol</b>	Numérico, opcional. Coordenada horizontal
<b>nWidth</b>	Numérico, opcional. Largura do objeto
<b>nHeight</b>	Numérico, opcional. Altura do objeto
<b>bFields</b>	Bloco de código, Lista de Campos
<b>aHeaders</b>	Vetor, Descrição dos campos para no cabeçalho
<b>aColSizes</b>	Vetor, Largura das colunas
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
<b>cField</b>	Caracter, opcional. Campos necessários ao filtro
<b>uValue1</b>	Indefinido, opcional. Inicio do intervalo para o filtro
<b>uValue2</b>	Indefinido, opcional. Fim do intervalo para o filtro
<b>bChange</b>	Bloco de código, opcional. Executado quando o item selecionado é alterado.
<b>bLDbClick</b>	Bloco de código, opcional. Executado quando acionado duplo click do botão esquerdo do mouse sobre o controle.
<b>bRClick</b>	Não utilizado
<b>oFont</b>	Objeto, opcional. Fonte
<b>oCursor</b>	Objeto, opcional. Tipo do Cursor
<b>nClrFore</b>	Numérico, opcional. Cor do texto da janela.
<b>nClrBack</b>	Numérico, opcional. Cor de fundo da janela.
<b>cMsg</b>	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
<b>IUpdate</b>	Não utilizado
<b>cAlias</b>	Caracter, opcional se objeto utilizado com Vetor, obrigatório se utilizado com Tabela
<b>IPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IDesign</b>	Não Utilizado
<b>bValid</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.

**Métodos auxiliares:**

**GoUp**

---

- Descrição:** Salta uma linha para cima.
- Sintaxe:** GoUp( )
- Parâmetros:**

Nenhum -

- Retorno:**

Nil

**GoDown**

---

- Descrição:** Salta uma linha para baixo.
- Sintaxe:** GoDown( )
- Parâmetros:**

Nenhum -

- Retorno:**

Nil

**GoTop**

---

- Descrição:** Salta para primeira linha.
- Sintaxe:** GoTop( )
- Parâmetros:**

Nenhum -

- Retorno:**

Nil

**GoBottom**

---

- Descrição:** Salta para ultima linha.
- Sintaxe:** GoBottom( )
- Parâmetros:**

Nenhum -

- Retorno:**

Nil

## RowCount

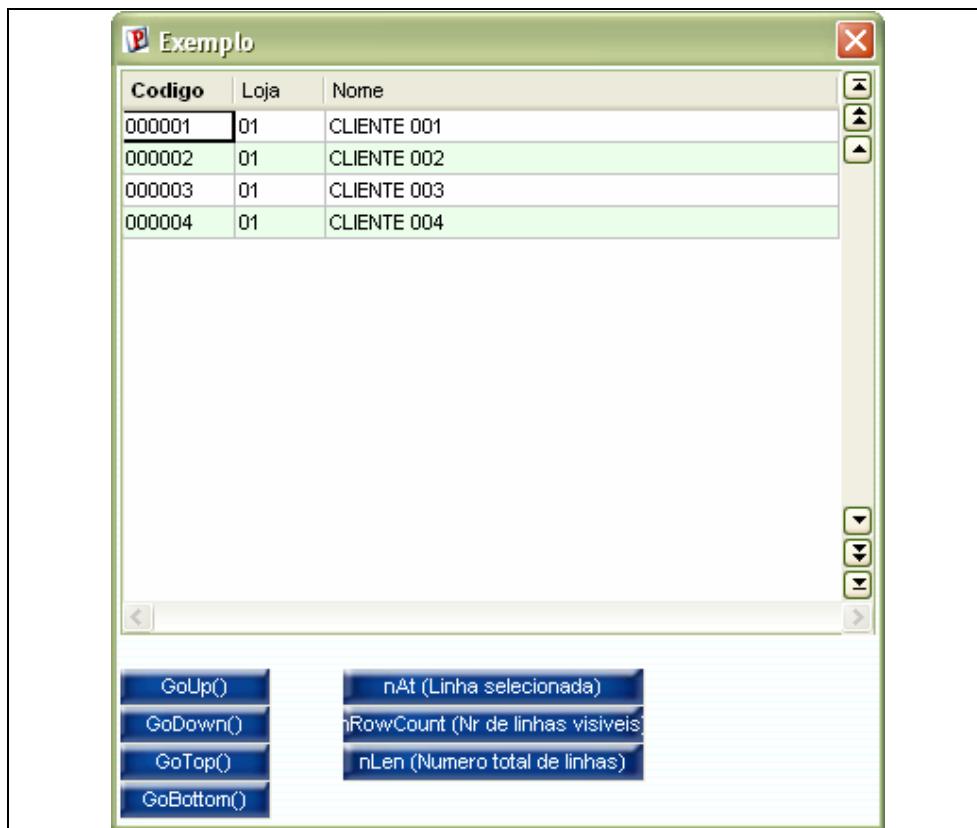
- Descrição:** Retorna numero de linhas visiveis.
- Sintaxe:** RowCount( )
- Parâmetros:**

Nenhum | -

- Retorno:**

Nil

**Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 302,402 PIXEL TITLE 'Exemplo'

        DbSelectArea('SA1')
        oBrowse := MsSelBr():New(
1,1,200,150,,,.oDlg,,,.F.,,'SA1',.T.,,.F.,,, )
        oBrowse:addColumn(TCColumn():New('Codigo',{||SA1->A1_COD
}.,,'LEFT',,.F.,.F.,,,.F.,))
        oBrowse:addColumn(TCColumn():New('Loja' ,{||SA1-
>A1_LOJA},,'LEFT',,.F.,.F.,,,.F.,))
        oBrowse:addColumn(TCColumn():New('Nome' ,{||SA1-
>A1_NOME},,'LEFT',,.F.,.F.,,,.F.,))

        // Principais comandos
        TButton():New(160,001,'GoUp()',oDlg,{|| oBrowse:GoUp()
},40,10,,,.T.)
        TButton():New(170,001,'GoDown()',oDlg,{|| oBrowse:GoDown()
},40,10,,,.T.)
        TButton():New(180,001,'GoTop()',oDlg,{|| oBrowse:GoTop()
},40,10,,,.T.)
        TButton():New(190,001,'GoBottom()' , oDlg,{|| oBrowse:GoBottom()
},40,10,,,.T.)
        TButton():New(160,060,'nAt (Linha selecionada)',oDlg,{|| 
Alert(oBrowse:nAt)},80,10,,,.T.)
        TButton():New(170,060,'nRowCount (Nr de linhas visiveis)',oDlg,;
{|| Alert(oBrowse:nRowCount()) },80,10,,,.T.)
        TButton():New(180,060, 'nLen (Numero total de linhas)' , oDlg,;
{|| Alert(oBrowse:nLen) },80,10,,,.T.)

        ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
.
```

---

## **MSWORKTIME()**

- Descrição:** Classe de objetos visuais do tipo controle - Barra de Período.

- Propriedades:**

<b>bChange</b>	Bloco de código. Executado ao disparar qualquer ação sobre o objeto.
----------------	--

- Construtor:** New([aoWnd], [anRow], [anCol], [anHeight], [anWidth], [nResolution], [cValue], [abWhen], [abChange])

- Parâmetros:**

<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.

<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>anHeight</b>	Numérico, opcional. Altura do botão em pixels.
<b>anWidth</b>	Numérico, opcional. Largura do botão em pixels.
<b>nResolution</b>	Numérico, Resolução a ser aplicada na grid do Calendário
<b>cValue</b>	Caracter, opcional. Descritivo
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>abChange</b>	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.



### Métodos auxiliares:

#### GetValue

---

- Descrição:** Retorna os item selecionados no formato "XX X XX".
- Sintaxe:** GetValue()
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Caracter</b>	Retorna os item selecionados no formato "XX X XX".
-----------------	--

#### GetInterTime

---

- Descrição:** Retorna periodo selecionado.
- Sintaxe:** GetInterTime()
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Caracter</b>	Retorna o período selecionada no formato "HH:MM:SS"
-----------------	---

#### SetResol

---

- Descrição:** Define a resolução da demonstração da barra de períodos.
- Sintaxe:** SetResol (nResolution)
- Parâmetros:**

<b>nResolution</b>	Resolução
--------------------	-----------

- Retorno:**

<b>Nil</b>	
------------	--

## **SetValue**

---

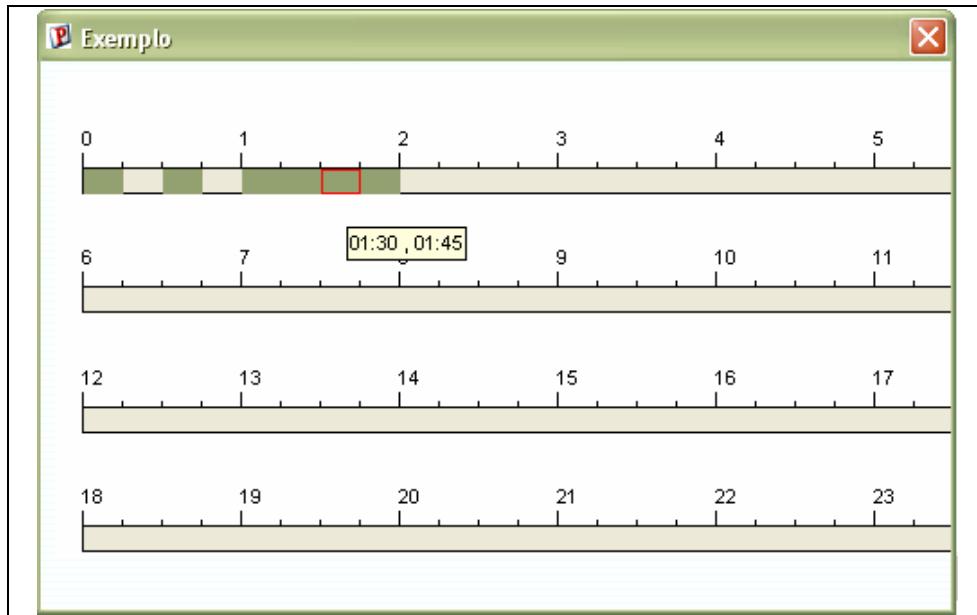
- ❑ **Descrição:** Atribui um determinado conteúdo para um objeto ou atributo de um objeto.
- ❑ **Sintaxe:** SetValue(xValue, cPicture)
- ❑ **Parâmetros:**

xValue	Valor que será atribuído ao objeto ou atributo do objeto.
cPicture	Picture de formação opcional para o conteúdo atribuído.

- ❑ **Retorno:**

Nenhum
--------

**Aparência:**



**Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    oMsWorkTime := MsWorkTime():New(oDlg,04,04,280,133,0,'',{||.T.},;
        {|oWorkTime|Alert('GetValue()':
            '+oWorkTime:getValue()+chr(13)+';
            'GetInterTime()':
            '+oWorkTime:getInterTime())} )
        oMsWorkTime:SetValue('X X XX X')
        ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **SBUTTON()**

---

- Descrição:** Classe de objetos visuais do tipo botão que Define o componente visual SButton, o qual permite a inclusão de botões de operação na tela da interface, os quais serão visualizados dependendo da interface do sistema ERP utilizada somente com um texto simples para sua identificação, ou com uma imagem (BitMap) pré-definido.

- Propriedades:**

<b>bAction</b>	Bloco de código. Executado ao precionar o botão esquerdo do mouse.
----------------	--

- Construtor:** New([nTop], [nLeft], [nType], [bAction], [oWnd], [lEnable], [cMsg], [bWhen])

- Parâmetros:**

Parâmetro	Tipo / Descrição
<b>nTop</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>nLeft</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>nType</b>	Numérico. Tipo do Botão
<b>bAction</b>	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.
<b>oWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>lEnable</b>	Logico, opcional. Habilita/Desabilita botão
<b>cMsg</b>	Caracter, Opicional. Descrição do tipo tooltip(hint) do item
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

- Métodos auxiliares:**

### **Create**

---

- Descrição:** Retorna Método construtor opcional da classe.

- Sintaxe:** Create()

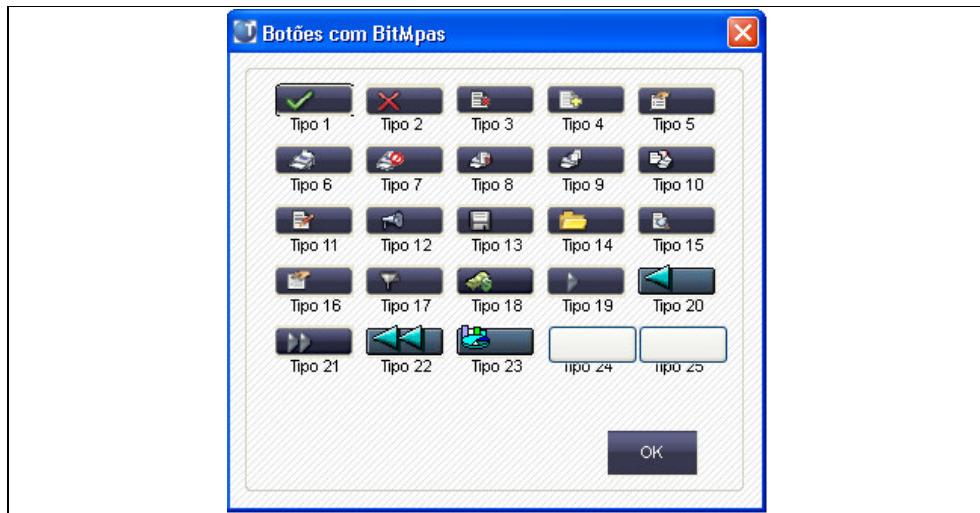
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Objeto</b>	Objeto da classe Sbutton com todos os atributos com conteúdos padrões.
---------------	--

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
       oSButton1 := SButton():New( 01,01,1,{||Alert('SButton
01')},oDlg,.T.,.)
       oSButton2 := SButton():New( 21,01,2,{||Alert('SButton
02')},oDlg,.T.,.)
       oSButton3 := SButton():New( 41,01,3,{||Alert('SButton
03')},oDlg,.T.,.)
       oSButton4 := SButton():New( 61,01,4,{||Alert('SButton
04')},oDlg,.T.,.)
    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```



*Anotações*

---

---

---

---

## **TBAR()**

---

- Descrição:** Classe de objetos visuais que permite a implementação de um componente do tipo barra de botões para a parte superior de uma janela previamente definida.

- Propriedades:**

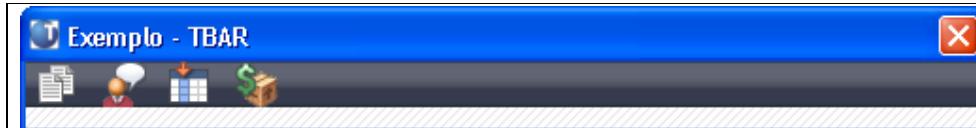
-	Herdados das classes superiores.
---	----------------------------------

- Construtor:**

- Parâmetros:**

<b>oWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>nBtnWidth</b>	Numérico, opcional. Largura do botão contido na barra
<b>nBtnHeight</b>	Numérico, opcional. Altura do botão contido na barra
<b>I3D</b>	Lógico, opcional. Define tipo da barra
<b>cMode</b>	Não utilizado.
<b>oCursor</b>	Objeto, opcional. Define Cursor ao posicionar o mouse sobre a barra.
<b>cResource</b>	Caracter, opcional. Imagem do recurso a ser inserido como fundo da barra.
<b>INoAutoAdjust</b>	Lógico.

- Aparência:**



## **Exemplo:**



Anotações

- 60 -

## TBITMAP()

---

- Descrição:** Classe de objetos visuais que permite a exibição de uma imagem.

**Propriedades:**

cResName	Caractere, Nome do resource
cBmpFile	Caractere, Nome do arquivo
ISstretch	Lógico, opcional. Strech da imagem
IAutoSize	Lógico, opcional. Tamanho automático
ITransparent	Lógico, opcional. Transparente

- Construtor:** New([anTop], [anLeft], [anWidth], [anHeight], [acResName], [acBmpFile], [alNoBorder], [aoWnd], [abLClicked], [abRClicked], [alScroll], [alStretch], [aoCursor], [acMsg], [alUpdate], [abWhen], [alPixel], [abValid], [alDesign], [alIsIcon], [alIsJpeg] )

**Parâmetros:**

<b>anTop</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anLeft</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>anWidth</b>	Numérico, opcional. Largura em pixels.
<b>anHeight</b>	Numérico, opcional. Altura em pixels.
<b>acResName</b>	Caractere, Nome do resource.
<b>acBmpFile</b>	Caractere, Nome do arquivo.
<b>alNoBorder</b>	Lógico, opcional. Exibe borda
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde a imagem deverá ser criada.
<b>abLClicked</b>	Bloco de código, opcional. Bloco que deverá ser acionado botão esquerdo do mouse for pressionado.
<b>abRClicked</b>	Bloco de código, opcional. Bloco que deverá ser acionado botão direito do mouse for pressionado.
<b>alScroll</b>	Lógico, opcional. Ativa o scroll
<b>alStretch</b>	Lógico, opcional. Strech da imagem
<b>aoCursor</b>	Objeto, opcional. Cursor a ser exibido
<b>acMsg</b>	Não utilizado
<b>alUpdate</b>	Não utilizado
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>alPixel</b>	Lógico, opcional. Utiliza coordenadas em pixel
<b>abValid</b>	Bloco de código, opcional. Bloco que deverá ser acionado na validação.
<b>alDesign</b>	Não utilizado
<b>alIsIcon</b>	Lógico, opcional. Ícone. Não utilizado a partir da versão Protheus 8
<b>alIsJpeg</b>	Lógico, opcional. Jpeg. Não utilizado a partir da versão Protheus 8

**Métodos auxiliares:**

**Create**

---

- Descrição:** Retorna Método construtor opcional da classe.
- Sintaxe:** Create()
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

<b>Objeto</b>	Objeto da classe TBar com todos os atributos com conteúdos padrões.
---------------	---

**SetBmp**

---

- Descrição:** Método para carregar uma imagem do repositório.
- Sintaxe:** SetBmp( acResName )
- Parâmetros:**

acResName	Caractere, Nome do resource
-----------	-----------------------------

- Retorno:**

Nenhum	-
--------	---

**Load**

---

- Descrição:** Método para carregar uma imagem do repositório ou local.
- Sintaxe:** Load( acResName, acBmpFile )
- Parâmetros:**

acResName	Caractere, Nome do resource.
acBmpFile	Caractere, Nome do arquivo.

- Retorno:**

Nenhum	-
--------	---

**Aparência:**



### **Exemplo:**

---

```
#include "protheus.ch"

User Function tBitmapTst()
Local oDlg
Local oBmp

DEFINE MSDIALOG oDlg TITLE '' FROM 0,0 TO 280,330 PIXEL
@ 10, 10 BITMAP oBmp RESOURCE 'totvs.bmp' SIZE 150,150 OF oDlg PIXEL
oBmp:lAutoSize := .T.

ACTIVATE MSDIALOG oDlg CENTERED

Return
.
```

### **TBROWSEBUTTON()**

---

- Descrição:** Classe de objetos visuais do tipo botão no formato padrão utilizado em browses da aplicação.

- Propriedades:**

<b>bAction</b>	Bloco de código. Executado ao precionar o botão esquerdo do mouse.
----------------	--

- Construtor:** New([nRow], [nCol], [cCaption], [oWnd], [bAction], [nWidth], [nHeight], [nHelpId], [oFont], [lDefault], [IPixel], [lDesign], [cMsg], [lUpdate], [bWhen], [bValid], [lCancel])

- Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>nCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>cCaption</b>	Caractere, opcional. Titulo do botão.
<b>oWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>bAction</b>	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.
<b>nWidth</b>	Numérico, opcional. Largura do botão em pixels
<b>nHeight</b>	Numérico, opcional. Altura do botão em pixels.
<b>nHelpId</b>	Reservado
<b>oFont</b>	Objeto, opcional. Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.
<b>lDefault</b>	Reservado
<b>IPixel</b>	Lógico, opcional. Se .T. considera as coordenadas passadas em pixels, se .F. (padrão) considera em caracteres.

<b>IDesign</b>	Reservado
<b>cMsg</b>	Reservado
<b>IUpdate</b>	Reservado
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>bValid</b>	Reservado
<b>ICancel</b>	Reservado

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    oTBrowseButton := TBrowseButton():New( 01,01,'TBrowseButton',oDlg,;
{|||Alert('TBrowseButton')},40,10,,, .F.,, .T.,, .F.,, .F.,,, )
    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```



*Anotações*

---



---



---



---

## TBTNBMP()

---

- Descrição:** Classe de objetos visuais do tipo botão, o qual permite que seja vinculada uma imagem ao controle.

**Propriedades:**

<b>bAction</b>	Bloco de código. Executado ao precionar o botão esquerdo do mouse.
----------------	--

- Construtor:** NewBar([cResName1], [cResName2], [cBmpFile1], [cBmpFile2], [cMsg], [bAction], [lGroup], [oWnd], [lAdjust], [bWhen], [cToolTip], [lPressed], [bDrop], [cAction], [nPos], [cPrompt], [oFont], [cResName3], [cBmpFile3], [lBorder])

**Parâmetros:**

<b>cResName1</b>	Caractere, Nome do resource
<b>cResName2</b>	Caractere, Nome do resource
<b>cBmpFile1</b>	Caractere, NÃO UTILIZADO
<b>cBmpFile2</b>	Caractere, NÃO UTILIZADO
<b>cMsg</b>	Caractere, Mensagem de Hint
<b>bAction</b>	Bloco de código. Ação executada
<b>lGroup</b>	Lógico. Define grupo
<b>oWnd</b>	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
<b>lAdjust</b>	Lógico, NÃO UTILIZADO
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>cToolTip</b>	Mensagem exibida quando objeto exibe seu tooltip.
<b>lPressed</b>	Não utilizado
<b>bDrop</b>	Não utilizado
<b>cAction</b>	Não utilizado
<b>nPos</b>	Não utilizado
<b>cPrompt</b>	Caracter, opcional. Caption do botão.
<b>oFont</b>	Objeto, opcional. Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do controle.
<b>cResName3</b>	Não utilizado
<b>cBmpFile3</b>	Não utilizado
<b>lBorder</b>	Não utilizado

**Métodos auxiliares:**

**LoadBitmaps**

---

- Descrição:** Atribui ao botão os bitmaps ou recursos para exibição.
- Sintaxe:** LoadBitmaps([cResName1], [cResName2], [cBmpFile1], [cBmpFile2], [cResName3], [cBmpFile3])

**Parâmetros:**

cResName1	Caractere, Nome do resource
cResName2	Caractere, Nome do resource
cBmpFile1	Caractere, Nome do arquivo BMP
cBmpFile2	Caractere, Nome do arquivo BMP
cResName3	Caractere, Nome do resource
cBmpFile3	Caractere, Nome do resource

**Retorno:**

Nenhum

**SetPopUpMenu**

---

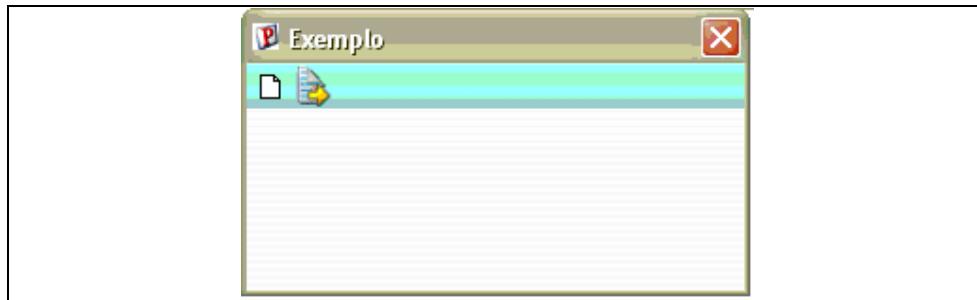
- Descrição:** Seta o objeto do tipo TMenu referente ao botão.
- Sintaxe:** SetPopupMenu(oMenu)
- Parâmetros:**

Nenhum -

**Retorno:**

Nenhum -

**Aparência:**



## **Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    oTBar := TBar():New( oDlg,25,32,.T.,,,.F. ) // Orig: 25,45
    oTBtnBmp1 := TBtnBmp() :NewBar('RPMNEW',,,,'Msg 01',;
        {||Alert('TBtnBmp
01')},.F.,oTBar,.T.,{||.T.},,.F.,,,1,,,,.T. )
    oTBtnBmp2 := TBtnBmp() :NewBar('copyuser',,,,'Msg 02',;
        {||Alert('TBtnBmp
02')},.F.,oTBar,.T.,{||.T.},,.F.,,,1,,,,.T. )

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **TBTNBMP2()**

- Descrição:** Classe de objetos visuais do tipo botão, o qual permite a exibição de uma imagem ou de um popup.

- Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([anTop], [anLeft], [anWidth], [anHeight], [acResName1], [acResName2], [acBmpFile1], [acBmpFile2],[abAction], [aoWnd], [acMsg], [abWhen], [IAdjuste], [IUpdate])

- Parâmetros:**

<b>anTop</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anLeft</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>anWidth</b>	Numérico, opcional. Largura em pixels.
<b>anHeight</b>	Numérico, opcional. Altura em pixels.
<b>acResName1</b>	Caractere, Nome do resource
<b>acResName2</b>	Caractere, NÃO UTILIZADO
<b>acBmpFile1</b>	Caractere, NÃO UTILIZADO
<b>acBmpFile2</b>	Caractere, NÃO UTILIZADO
<b>abAction</b>	Bloco de código. Ação executada
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
<b>acMsg</b>	Caractere, Mensagem de Hint
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IAdjust</b>	Lógico, NÃO UTILIZADO
<b>IUpdate</b>	Lógico, NÃO UTILIZADO

- Métodos auxiliares:**

## **LoadBitmaps**

---

- Descrição:** Atribui ao botão os bitmaps ou recursos para exibição.
- Sintaxe:** LoadBitmaps([cResName1], [cResName2], [cBmpFile1], [cBmpFile2], [cResName3], [cBmpFile3])

- Parâmetros:**

<b>cResName1</b>	Caractere, Nome do resource
<b>cResName2</b>	Caractere, Nome do resource
<b>cBmpFile1</b>	Caractere, Nome do arquivo BMP
<b>cBmpFile2</b>	Caractere, Nome do arquivo BMP
<b>cResName3</b>	Caractere, Nome do resource
<b>cBmpFile3</b>	Caractere, Nome do resource

- Retorno:**

<b>Nenhum</b>
---------------

## **SetPopUpMenu**

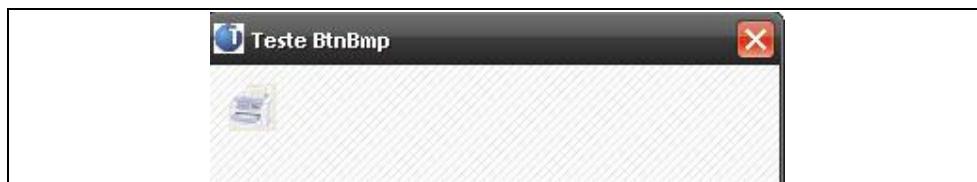
---

- Descrição:** Seta o objeto do tipo TMenu referente ao botão.
- Sintaxe:** SetPopupMenu(oMenu)
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Aparência:**

## **Exemplo:**

---

```
Local oBtn := TBtnBmp2():New( 10, 10, 25, 25, 'printer_r2' , , , , , oDlg,  
, , )  
Exemplo 2 (Popup)  
#INCLUDE 'PROTHEUS.CH'  
User Function BtnBmpPopUp  
Local oDlg  
Local oBtn  
DEFINE MSDIALOG oDlg TITLE '' FROM 0,0 TO 100,200 PIXEL  
oBtn := TBtnBmp2():New( 10, 10, 13, 13, 'IBF-MENU.BMP' , 'IBF-MENU.BMP' ,  
, , , oDlg, , , .T.)  
oBtn:SetPopupmenu(TShowMenu())  
ACTIVATE MSDIALOG oDlg CENTERED  
Return  
  
*****  
Static Function TShowMenu()  
Local oMenu  
oMenu := TMenu():New(0,0,0,0,.T.)  
oMenu:Add(TMenuItem():New('Detalhes', 'Detalhes', , , , , , , , , , , , , , , , .T.))  
oMenu:Add(TMenuItem():New('Add Info', 'Add Info', , , , , , , , , , , , , , , , .T.))  
Return oMenu
```

## **TBUTTON()**

---

- Descrição:** Classe de objetos visuais do tipo botão, o qual permite a utilização de texto para sua identificação.

**Propriedades:**

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

- Construtor:** New([anRow], [anCol], [acCaption], [aoWnd], [abAction], [anWidth], [anHeight], [nPar8], [aoFont], [lPar10], [alPixel],[lPar12],[cPar13], [lPar14], [abWhen], [bPar16], [lPar17])

**Parâmetros:**

Parâmetro	Tipo / Descrição
<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>acCaption</b>	Caractere, opcional. Titulo do botão.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>abAction</b>	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.
<b>anWidth</b>	Numérico, opcional. Largura do botão em pixels.
<b>anHeight</b>	Numérico, opcional. Altura do botão em pixels.
<b>nPar8</b>	Reservado.
<b>aoFont</b>	Objeto, opcional. Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.

<b>IPar10</b>	Reservado.
<b>alPixel</b>	Lógico, opcional. Se .T. considera as coordenadas passadas em pixels, se .F. (padrão) considera em caracteres.
<b>IPar12</b>	Reservado.
<b>cPar13</b>	Reservado.
<b>IPar14</b>	Reservado.
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>bPar16</b>	Reservado.
<b>IPar17</b>	Reservado.

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function TesteGet()
Local oDlg, oButton, oCombo, cCombo
aItems:= {'item1','item2','item3'}
cCombo:= aItems[2]
DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu Combo'
  oCombo:= tComboBox():New(10,10,{|u|if(PCount()>0,cCombo:=u,cCombo)},;
    aItems,100,20,oDlg,,{||MsgStop('Mudou item')},,.T.,.,.,,'cCombo')

  // Botão para fechar a janela
  oButton:=tButton():New(30,10,'fechar',oDlg,{||oDlg:End()},100,20,,,.T.)
ACTIVATE MSDIALOG oDlg CENTERED
MsgStop( 'O valor é '+cCombo )
Return NIL
```

## **TCBROWSE()**

---

- Descrição:** Classe de objetos visuais do tipo Grid.

- Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>nAt</b>	Linha atualmente selecionada / posicionada no objeto
<b>nLen</b>	Número total de linhas do objeto

- Construtor:** New([nRow], [nCol], [nWidth], [nHeighth],[bFields], [aHeaders], [aColSizes], [oDlg], [cField], [uValue1], [uValue2], [uChange],[{|nRow,nCol,nFlags|}[uLDbClick]]], [|{|nRow,nCol,nFlags|}[uRClick]]], [oFont], [oCursor], [nClrFore], [nClrBack], [cMsg], [IUpdate], [cAlias], [IPixel], [{uWhen}], [IDesign], [bValid], [IHScroll], [IVScroll]])

- Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical
<b>nCol</b>	Numérico, opcional. Coordenada horizontal
<b>nWidth</b>	Numérico, opcional. Largura do objeto
<b>nHeight</b>	Numérico, opcional. Altura do objeto
<b>bFields</b>	Bloco de código, Lista de Campos
<b>aHeaders</b>	Vetor, Descrição dos campos para no cabeçalho
<b>aColSizes</b>	Vetor, Largura das colunas
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
<b>cField</b>	Caracter, opcional. Campos necessários ao filtro
<b>uValue1</b>	Indefinido, opcional. Inicio do intervalo para o filtro
<b>uValue2</b>	Indefinido, opcional. Fim do intervalo para o filtro
<b>bChange</b>	Bloco de código, opcional. Executado quando o item selecionado é alterado.
<b>bLDbClick</b>	Bloco de código, opcional. Executado quando acionado duplo click do botão esquerdo do mouse sobre o controle.
<b>bRClick</b>	Não utilizado
<b>oFont</b>	Objeto, opcional. Fonte
<b>oCursor</b>	Objeto, opcional. Tipo do Cursor
<b>nClrFore</b>	Numérico, opcional. Cor do texto da janela.
<b>nClrBack</b>	Numérico, opcional. Cor de fundo da janela.
<b>cMsg</b>	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
<b>IUpdate</b>	Não utilizado
<b>cAlias</b>	Caracter, opcional se objeto utilizado com Vetor, obrigatório se utilizado com Tabela
<b>IPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IDesign</b>	Não Utilizado
<b>bValid</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.
<b>IHScroll</b>	Lógico, opcional. Se .T., habilita barra de rolagem horizontal.
<b>IVScroll</b>	Lógico, opcional. Se .T., habilita barra de rolagem vertical.

**Métodos auxiliares:**

#### GoUp

- Descrição:** Salta uma linha para cima.
- Sintaxe:** GoUp( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nil</b>
------------

#### GoDown

- Descrição:** Salta uma linha para baixo.
- Sintaxe:** GoDown( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nil</b>
------------

#### GoTop

- Descrição:** Salta para primeira linha.
- Sintaxe:** GoTop( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Nil</b>
------------

#### GoBottom

- Descrição:** Salta para ultima linha.
- Sintaxe:** GoBottom( )
- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

**Nil**

### RowCount

- Descrição:** Retorna numero de linhas visiveis.
- Sintaxe:** RowCount( )
- Parâmetros:**

**Nenhum**

- Retorno:**

**Nil**

### Aparência:

**Exemplo da TCBrowse**

Codigo	Descrição	Valor
0000000001	Descrição do Produto 001	1.001,22
0000000002	EDITADA PELO LEDITCELL	1.002,22
0000000003	Descrição do Produto 003	1.003,22
0000000004	Descrição do Produto 004	1.004,22
0000000005	Descrição do Produto 005	1.005,22
0000000006	Descrição do Produto 006	1.006,22
0000000007	Descrição do Produto 007	1.007,22
0000000008	Descrição do Produto 008	1.008,22
0000000009	Descrição do Produto 009	1.009,22
0000000010	Descrição do Produto 010	1.010,22
0000000011	Descrição do Produto 011	1.011,22
0000000012	Descrição do Produto 012	1.012,22
0000000013	Descrição do Produto 013	1.013,22
0000000014	Descrição do Produto 014	1.014,22
0000000015	Descrição do Produto 015	1.015,22
0000000016	Descrição do Produto 016	1.016,22
0000000017	Descrição do Produto 017	1.017,22
0000000018	Descrição do Produto 018	1.018,22
0000000019	Descrição do Produto 019	1.019,22
0000000020	Descrição do Produto 020	1.020,22
0000000021	Descrição do Produto 021	1.021,22

GoUp()

nAt (Linha selecionada)

GoDown()

nRowCount (Nr de linhas visiveis)

GoTop()

nLen (Número total de linhas)

GoBottom()

lEditCell (Edita a célula)

## **Exemplo:**

```
#include 'protheus.ch'
user function TcBrowse_EX()
Local oOK    := LoadBitmap(GetResources(),'br_verde')
Local oNO    := LoadBitmap(GetResources(),'br_vermelho')
Local aList := {} // Vetor com elementos do Browse
Local nX
    // Cria Vetor para teste
for nX := 1 to 100
    aListAux := { .T., strzero(nX,10), 'Descrição do Produto '+;
        strzero(nX,3), 1000.22+nX}
    aadd(aList, aListAux)
next

DEFINE MSDIALOG oDlg FROM 0,0 TO 520,600 PIXEL TITLE 'Exemplo da TCBrowse'
    // Cria objeto de fonte que sera usado na Browse
    Define Font oFont Name 'Courier New' Size 0, -12
    // Cria Browse
    oList := TCBrowse():New( 01 , 01, 300, 200,,,
        '','Codigo','Descrição','Valor'),{20,50,50,50},,
        oDlg,,,{||},,oFont,,,.F.,,T.,,F.,, )
    // Seta o vetor a ser utilizado
    oList:SetArray(aList)
    // Monta a linha a ser exibida no Browse
    oList:bLine := {||{ If(aList[oList:nAt,01],oOK,oNO),;
        aList[oList:nAt,02],;
        aList[oList:nAt,03],;
        Transform(aList[oList:nAt,04],'@E 99,999,999,999.99') } }
    // Evento de DuploClick (troca o valor do primeiro elemento do Vetor)
    oList:bLDbClick := {|| aList[oList:nAt][1] :=;
        !aList[oList:nAt][1],oList:DrawSelect() }
    // Principais comandos
    oBtn := TButton():New( 210, 001,'GoUp()', oDlg,{||oList:GoUp()},;
        40, 010,,,.F.,,T.,,F.,,F.,, )
    oBtn := TButton():New( 220, 001,'GoDown()', oDlg,{||oList:GoDown()},;
        40, 010,,,.F.,,T.,,F.,,F.,, )
    oBtn := TButton():New( 230, 001,'GoTop()', oDlg,{||oList:GoTop()},;
        40, 010,,,.F.,,T.,,F.,,F.,, )
    oBtn := TButton():New( 240, 001,'GoBottom()', oDlg,{||oList:GoBottom()},;
        40, 010,,,.F.,,T.,,F.,,F.,, )
    oBtn := TButton():New( 210, 060, 'nAt (Linha selecionada)' ,;
        oDlg,{|| Alert(oList:nAt)},;
        90, 010,,,.F.,,T.,,F.,,F.,, )
    oBtn := TButton():New( 220, 060, 'nRowCount (Nr de linhas visiveis)',;
        oDlg,{|| Alert(oList:nRowCount()) }, 90, 010,,,.F.,,T.,,F.,,F.,,F., )
    oBtn := TButton():New( 230, 060, 'nLen (Numero total de linhas)', oDlg,;
        {|| Alert(oList:nLen) }, 90, 010,,,.F.,,T.,,F.,,F.,,F., )
    oBtn := TButton():New( 240, 060, 'lEditCell (Edita a celula)', oDlg,;
        {|| lEditCell(@aList,oList,'@!',3) }, 90, 010,,,.F.,,T.,,F.,,F.,,F., )
ACTIVATE MSDIALOG oDlg CENTERED
return
```

## TCHECKBOX()

---

- Descrição:** Classe de objetos visuais do tipo controle - CheckBox.

- Propriedades:**

<b>bLClicked</b>	Bloco de código disparado ao clique do mouse no objeto
<b>bSetGet</b>	Bloco de código disparado na mudança de item selecionado, responsável pela mudança de valor da variável numérica que indica o item selecionado.
<b>bWhen</b>	Bloco de código recebe um lógico e de True permite alteração, se False não permite.
<b>bValid</b>	Bloco de código executado na saída do objeto.

- Construtor:** New([nRow], [nCol], [cCaption], [bSetGet], [oDlg], [nWidth], [nHeight], [aHelpIds], [bLClicked], [oFont], [bValid], [nClrText], [nClrPane], [IDesign], [IPixel], [cMsg], [IUpdate], [bWhen])

- Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical
<b>nCol</b>	Numérico, opcional. Coordenada horizontal
<b>cCaption</b>	Caracter, Texto descritivo
<b>bSetGet</b>	Code-block, opcional, Responsável pela setagem de valor
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
<b>nWidth</b>	Numérico, opcional. Largura do objeto
<b>nHeight</b>	Numérico, opcional. Altura do objeto
<b>aHelpIds</b>	Não utilizado
<b>bLClicked</b>	Bloco de código, opcional. Executado ao clique do mouse.
<b>oFont</b>	Objeto, opcional. Fonte
<b>bValid</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.
<b>nClrText</b>	Numérico, opcional. Cor do texto da janela.
<b>nClrPane</b>	Numérico, opcional. Cor de fundo da janela.
<b>IDesign</b>	Não utilizado
<b>IPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>cMsg</b>	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
<b>IUpdate</b>	Não utilizado
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
user function TCheckBox()
    DEFINE MSDIALOG oDlg FROM 0,0 TO 270,400 PIXEL TITLE 'Exemplo da
TCBrowse'
    lCheck1 := .T.
    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....)

    oCheck4 :=TCheckBox():New(31,01,'CheckBox 004',,oDlg,
100,210,,.....T....)
    oCheck5 := TCheckBox():New(41,01,'CheckBox
005',,oDlg,100,210,,.....T....)

        // Seta Eventos do primeiro Check
    oCheck1:bSetGet := {|| lCheck1 }
    oCheck1:bLClicked := {|| lCheck1:=!lCheck1 }
    oCheck1:bWhen      := {|| .T. }
    oCheck1:bValid     := {|| Alert('bValid') }
    // Principais comandos
    oBtn  := TButton():New( 060, 001, 'Retorna estado do CheckBox
001',;
                           oDlg,{|| Alert(lCheck1) }, 120,
010,,,.F.,.T.,.F.,.,.F.,.,.F. )
    ACTIVATE MSDIALOG oDlg CENTERED
return
```

## **TCOLORTRIANGLE()**

---

- Descrição:** Classe de objetos visuais do tipo Paleta de Cores.

- Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([anRow], [anCol], [aoWnd], [anWidth], [anHeight] )

- Parâmetros:**

<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde a paleta deverá ser criada.
<b>anWidth</b>	Numérico, opcional. Largura da paleta em pixels.
<b>anHeight</b>	Numérico, opcional. Altura da paleta em pixels.

- Métodos auxiliares:**

### **Create**

---

- Descrição:** Método construtor opcional da classe.

- Sintaxe:** Create(aoWnd)

- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde a paleta deverá ser criada.
-------	---

- Retorno:**

<b>Objeto</b>	Objeto do tipo TColorTriangule com os atributos definidos com conteúdo padrão.
---------------	--

### **RetColor**

---

- Descrição:** Retorna o RGB da cor selecionada

- Sintaxe:** RetColor ( )

- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

<b>Numérico</b>	Valor que representa do RGB da cor.
-----------------	-------------------------------------

## **SetColorIni**

- Descrição:** Define a cor inicial selecionada para o controle.
- Sintaxe:** **SetColorIni** (nColor )
- Parâmetros:**

nColor	Valor da cor inicial no padrão RGB.
--------	-------------------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SetColor**

- Descrição:** Define a cor corrente.
- Sintaxe:** **SetColor** (nColor )
- Parâmetros:**

nColor	Valor da cor inicial no padrão RGB.
--------	-------------------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SetSizeTriangule**

- Descrição:** Define o tamanho do triângulo de configuração das cores.
- Sintaxe:** **SetSizeTriangule** (nWidth, nHeight)
- Parâmetros:**

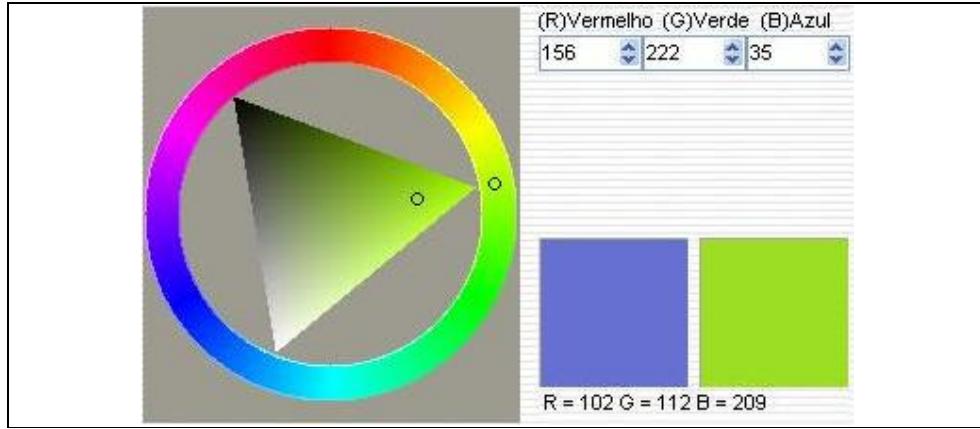
nWidth	Numérico. Largura
nHeight	Numérico. Altura

- Retorno:**

<b>Nenhum</b>	-
---------------	---



### **Aparência:**



## **Exemplo:**

---

```
#include "protheus.ch"

User Function tColor()
Local oDlg, oColorT
DEFINE MSDIALOG oDlg FROM 0,0 TO 500,600 PIXEL TITLE "Cores"
// Usando o método create
oColorT := tColorTriangle():Create( oDlg )
oColorT:SetColorIni( nColorIni )
ACTIVATE MSDIALOG oDlg CENTERED
Return Nil
```

## **TCOMBOBOX()**

---

- Descrição:** Classe de objetos visuais do tipo tComboBox, a qual cria uma entrada de dados com múltipla escolha com item definido em uma lista vertical, acionada 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"} ou b) Indexada, exemplo: {"a=item1","b=item2", ... , "n=itemN"}.
<b>nAt</b>	Numérico. Posição do item selecionado.

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

- Parâmetros:**

<b>Parâmetro</b>	
<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>abSetGet</b>	Bloco de código, opcional. Bloco de código no formato { u  if( Pcount( )>0, <var>:= u, <var> ) } que o controle utiliza para atualizar a variável <var>. <var> deve ser tipo caracter. 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.
<b>anItems</b>	Array, opcional. Lista de items, 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"}.
<b>anWidth</b>	Numérico, opcional. Largura do controle em pixels.

<b>anHeight</b>	Numérico, opcional. Altura do controle em pixels.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o controle será criado.
<b>nPar8</b>	Reservado.
<b>abChange</b>	Bloco de código, opcional. Executado quando o controle modifica o item selecionado.
<b>abValid</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.
<b>anClrBack</b>	Numérico, opcional. Cor de fundo do controle.
<b>anClrText</b>	Numérico, opcional. Cor do texto do controle.
<b>alPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>aoFont</b>	Objeto, opcional. Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do controle.
<b>cPar15</b>	Reservado.
<b>IPar16</b>	Reservado.
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IPar18</b>	Reservado.
<b>aPar19</b>	Reservado.
<b>bPar20</b>	Reservado.
<b>cPar21</b>	Reservado.
<b>acReadVar</b>	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( ).



### Métodos auxiliares:

#### Select

---

- Descrição:** Muda o item selecionado no combobox.
- Sintaxe:** Select( [anItem] )
- Parâmetros:**

anItem	Numérico, opcional. Posição do item a ser selecionado.
--------	--

- Retorno:**

Nenhum	-
--------	---

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function TesteGet()
Local oDlg, oButton, oCombo, cCombo
aItems:= {'item1','item2','item3'}
cCombo:= aItems[2]
DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu Combo'
oCombo:= tComboBox():New(10,10,{|u|if(PCount()>0,cCombo:=u,cCombo)},;
aItems,100,20,oDlg,,{|MsgStop('Mudou item')},,.T.,,'cCombo')

// Botão para fechar a janela
oButton:=tButton():New(30,10,'fechar',oDlg,{||oDlg:End()},100,20,..T.)
ACTIVATE MSDIALOG oDlg CENTERED
MsgStop( 'O valor é '+cCombo )
Return NIL
```

## **TFOLDER()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Folder.

- Propriedades:**

<b>aPrompts</b>	Array, Titulo dos folders
<b>aDialogs</b>	Array, Nome dos diálogos
<b>nOption</b>	Numérico, Folder selecionado
<b>bSetOption</b>	Bloco de código, Executado na seleção do folder

- Construtor:** New([anTop], [anLeft], [aPrompts], [aDialogs], [aoWnd], [anOption], [anClrFore], [anClrBack], [alPixel], [alDesign], [anWidth], [anHeight], [acMsg], [alAllWidth])

- Parâmetros:**

<b>anTop</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anLeft</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>aPrompts</b>	Array, Titulo dos folders
<b>aDialogs</b>	Array, Nome dos diálogos
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
<b>anOption</b>	Numérico, opcional. Folder selecionado
<b>anClrFore</b>	Numérico, opcional. Cor de frente
<b>anClrBack</b>	Numérico, opcional. Cor de fundo
<b>alPixel</b>	Lógico, opcional. Utiliza coordenadas em pixel
<b>alDesign</b>	Lógico, opcional. NÃO USADO
<b>anWidth</b>	Numérico, opcional. Largura em pixels.
<b>anHeight</b>	Numérico, opcional. Altura em pixels.
<b>acMsg</b>	Caractere, Mensagem de Hint
<b>alAllWidth</b>	Lógico, opcional. NÃO USADO

- Métodos auxiliares:**

### **SetOption**

---

- Descrição:** Seleciona folder desejado

- Sintaxe:** SetOption( nOption )

- Parâmetros:**

<b>nOption</b>	Numérico. Folder a ser selecionado
----------------	------------------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## AddItem

---

- Descrição:** Insere um folder
- Sintaxe:** AddItem ( cItem, lVisible )
- Parâmetros:**

cItem	Caractere. Título do Folder
lVisible	Lógico. Visível

- Retorno:**

Nenhum	-
--------	---

## aEnable

---

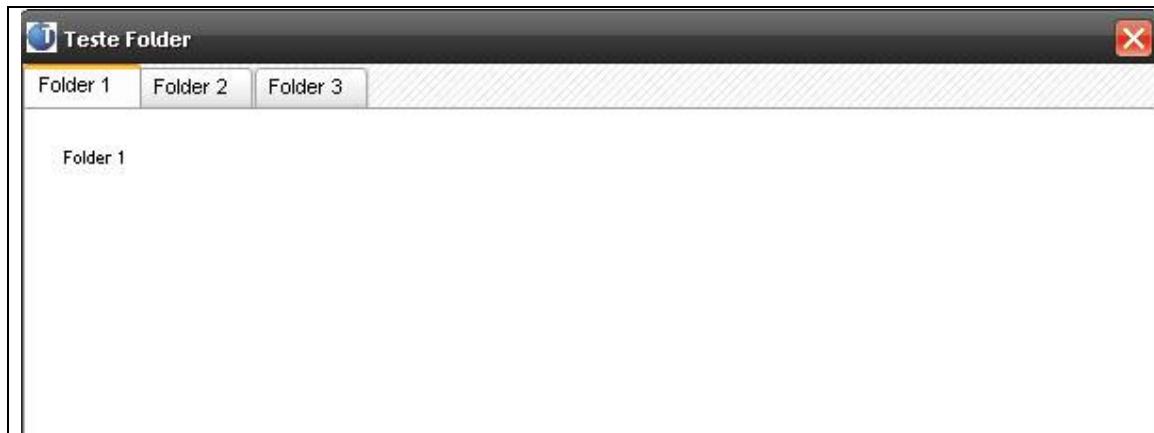
- Descrição:** Habilita/Desabilita um folder
- Sintaxe:** aEnable( nItem, lEnable )
- Parâmetros:**

nItem	Numérico. Folder para habilitar/desabilitar
lEnable	Lógico. Habilita/Desabilita

- Retorno:**

Nenhum	-
--------	---

- Aparência:**



## **Exemplo:**

---

```
Local oFolder
Local aFolder := { 'Folder 1', 'Folder 2', 'Folder 3' }

oFolder := TFolder():New( 0, 0, aFolder, aFolder, oDlg,,,, .T., , 200,;
200 )
```

## **TGET()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tGet, a qual cria um controle que armazena ou altera o conteúdo de uma variável através de digitação. O conteúdo da variável só é modificado quando o controle perde o foco de edição para outro controle.

**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.

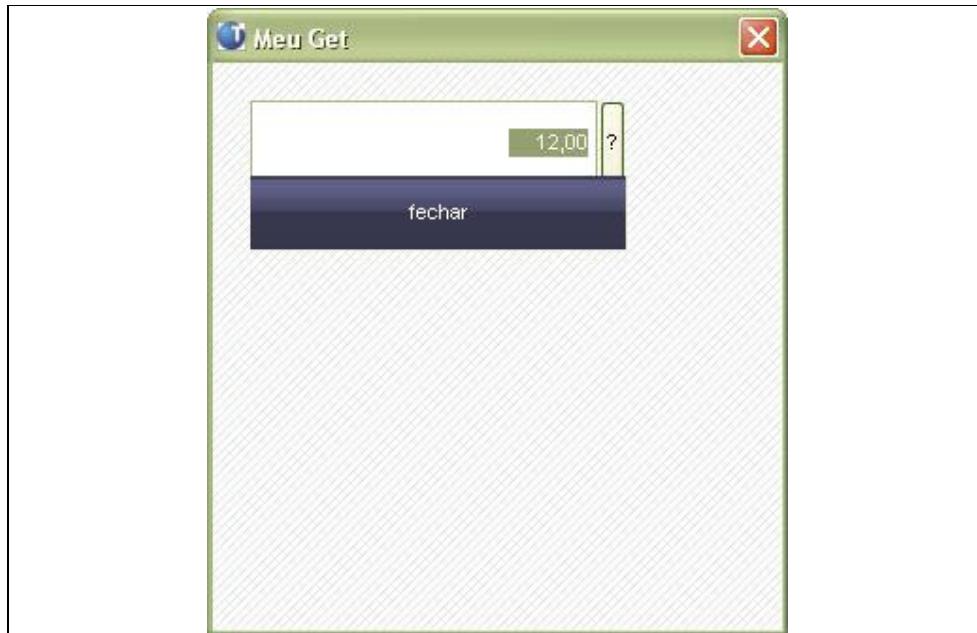
- Construtor:** New([anRow], [anCol], [abSetGet], [aoWnd], [anWidth], [anHeight], [acPict], [abValid], [anClrFore], [anClrBack], [aoFont], [IPar12], [oPar13], [alPixel], [cPar15], [IPar16], [abWhen], [IPar18], [IPar19], [abChange], [alReadOnly], [alPassword], [cPar23], [acReadVar], [cPar25], [IPar26],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

**Parâmetros:**

<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>abSetGet</b>	Bloco de código, opcional. Bloco de código no formato { u  if( Pcount( )>0, <var>:= u, <var> ) } que o controle utiliza para atualizar a variável <var>. <var> deve ser tipo caracter, numérico ou data.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o controle será criado.
<b>anWidth</b>	Numérico, opcional. Largura do controle em pixels.
<b>anHeight</b>	Numérico, opcional. Altura do controle em pixels.
<b>acPict</b>	Caractere, opcional. Máscara de formatação do conteúdo a ser exibido.
<b>abValid</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.
<b>anClrFore</b>	Numérico, opcional. Cor de fundo do controle.
<b>anClrBack</b>	Numérico, opcional. Cor do texto do controle.
<b>aoFont</b>	Objeto, opcional. Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do

	controle.
<b>IPar12</b>	Reservado.
<b>oPar13</b>	Reservado.
<b>alPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>cPar15</b>	Reservado.
<b>IPar16</b>	Reservado.
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IPar18</b>	Reservado.
<b>IPar19</b>	Reservado.
<b>abChange</b>	Bloco de código, opcional. Executado quando o controle modifica o valor da variável associada.
<b>alReadOnly</b>	Lógico, opcional. Se .T. o controle não poderá ser editado.
<b>alPassword</b>	Lógico, opcional. Se .T. o controle exibirá asteriscos "*" no lugar dos caracteres exibidos pelo controle para simular entrada de senha.
<b>cPar23</b>	Reservado.
<b>acReadVar</b>	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( ).
<b>cPar25</b>	Reservado.
<b>IPar26</b>	Reservado.
<b>nPar27</b>	Reservado.
<b>IPar28</b>	Reservado.

**Aparência:**



## **Exemplo:**

## TGROUP()

- Descrição:** Classe de objetos visuais do tipo painel – tGroup, a qual cria um painel onde controles visuais podem ser agrupados ou classificados. Neste painel é criada uma borda com título em volta dos controles agrupados.

**Propriedades:**

**Propriedades:**

- Herdadas das classes superiores.

- Construtor:** New([anTop], [anLeft], [anBottom], [anRight], [acCaption],  
[aoWnd], [anClrText], [anClrPanel], [alPixel], [lPar10])

**Parâmetros:**

<b>anTop</b>	Numérico, opcional. Coordenada vertical superior em pixels ou caracteres.
<b>anLeft</b>	Numérico, opcional. Coordenada horizontal esquerda em pixels ou caracteres.
<b>anBottom</b>	Numérico, opcional. Coordenada vertical inferior em pixels ou caracteres.
<b>anRight</b>	Numérico, opcional. Coordenada horizontal direita em pixels ou caracteres.
<b>acCaption</b>	Caractere, opcional. Título do grupo.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o controle será criado.
<b>anClrText</b>	Numérico, opcional. Cor do texto.
<b>anClrPane</b>	Numérico, opcional. Cor do fundo.
<b>alPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>IPar10</b>	Reservado.

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg, oButton, oCombo, cCombo, cGet1:='Teste'
DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu test'
oGroup:= tGroup():New(10,10,130,130,'grupo de gets',oDlg,,,T.)
@ 18,16 MSGET oGet1 VAR cGet1 SIZE 100,10 OF oGroup PIXEL
@ 38,16 MSGET oGet2 VAR cGet1 SIZE 100,10 OF oGroup PIXEL
ACTIVATE MSDIALOG oDlg CENTERED
```



*Anotações*

## **THBUTTON()**

---

- Descrição:** Classe de objetos visuais do tipo botão com hiperlink.

- Propriedades:**

bAction	Bloco de código. Executado quando o botão é pressionado.
---------	--

- Construtor:** New([anRow], [anCol], [acCaption], [aoWnd], [abAction], [anWidth], [anHeight], [aoFont], [abWhen])

- Parâmetros:**

Parâmetro	Tipo / Descrição
<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>acCaption</b>	Caractere, opcional. Título do botão.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>abAction</b>	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.
<b>anWidth</b>	Numérico, opcional. Largura do botão em pixels.
<b>anHeight</b>	Numérico, opcional. Altura do botão em pixels.
<b>aoFont</b>	Objeto, opcional. Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

- Métodos auxiliares:**

### **Create**

---

- Descrição:** Método construtor opcional da classe.

- Sintaxe:** Create(aoWnd)

- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde a paleta deverá ser criada.
-------	---

- Retorno:**

<b>Objeto</b>	Objeto do tipo THButton com os atributos definidos com conteúdo padrão.
---------------	---

**Aparência:**



**Exemplo:**

```
#include "protheus.ch"
#include "hbutton.ch"
User Function MyhBtn()
Local oDlg, ohBtn
DEFINE MSDIALOG oDlg FROM 0,0 TO 500,600 PIXEL TITLE "Meu hButton"
// Usando o método create
ohBtn:= tHButton():Create( oDlg )
ohBtn:nTop      := 100
ohBtn:nLeft     := 10
ohBtn:nWidth    := 100
ohBtn:nHeight   := 30
ohBtn:cCaption := 'hButton'
ohBtn:blClicked := { || MsgStop( 'Cliquei' ) }
// Usando o command
@ 200,100 HBUTTON ohBtn PROMPT 'Exemplo hButton' SIZE 100,30 ACTION
MsgStop('Cliquei') OF oDlg MESSAGE 'hButton'
ACTIVATE MSDIALOG oDlg CENTERED
Return Nil
```

**TIBROWSER()**

- Descrição:** Classe de objetos visuais do tipo Página de Internet, sendo necessário incluir a clausula BrowserEnabled=1 no Config do Remote.INI

**Propriedades:**

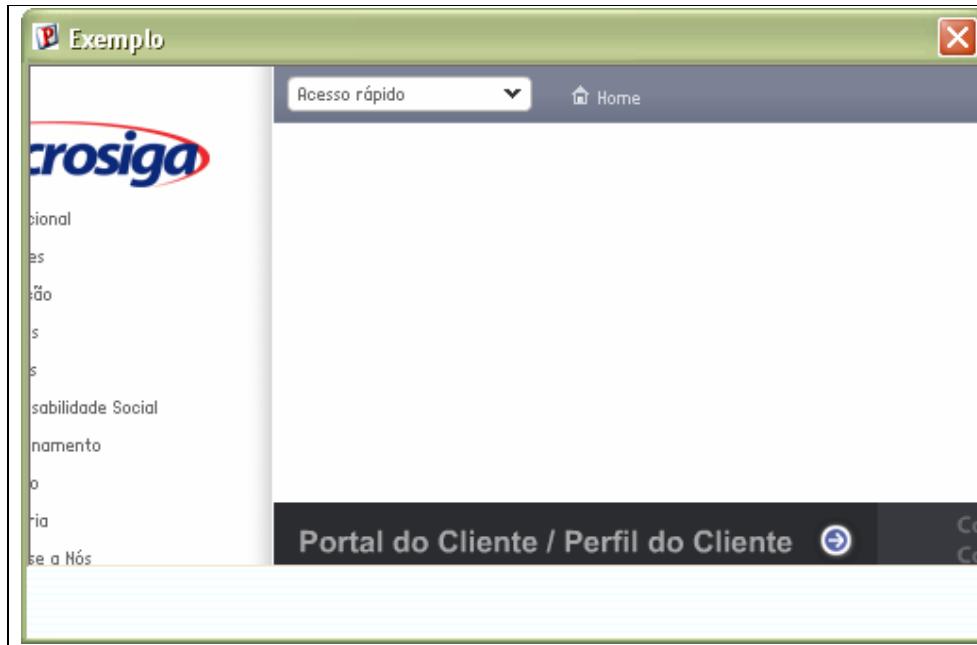
-	Herdadas da classe superior
---	-----------------------------

- Construtor:** New([nRow], [nCol], [nWidth], [nHeight], [cPage], [oWnd])

**Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>nCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres
<b>nWidth</b>	Numérico, opcional. Largura em pixels
<b>nHeight</b>	Numérico, opcional. Altura em pixels.
<b>cPage</b>	Caracter. Enredeço URL da página de Internet
<b>oWnd</b>	Objeto, opcional. Janela ou controle onde a botão deverá ser criado

**Aparência:**



### **Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'

    // Habilitar no Remote(Config) -> BrowserEnabled=1
    oTIBrowser:=
    TIBrowser():New(0,0,306,134,'http://www.microsiga.com.br',oDlg )

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```



### ***Anotações***

---

---

---

---

## **TLISTBOX()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tListbox, a qual cria uma janela com itens selecionáveis e barra de rolagem. Ao selecionar um item, uma variável é atualizada com o conteúdo do item selecionado.

**Propriedades:**

<b>nAt</b>	Numérico. Posição do item selecionado.
<b>aItems</b>	Array de items caracteres. Lista do itens selecionáveis.

- Construtor:** New([anRow], [anCol], [abSetGet], [aaItems], [anWidth], [anHeigth], [abChange], [aoWnd], [abValid], [anClrFore], [anClrBack], [alPixel], [IPar13], [abLDBLClick], [aoFont], [cPar16], [IPar17], [abWhen], [aPar19], [bPar20], [IPar21], [IPar22], [abRightClick] )

**Parâmetros:**

<b>anRow</b>	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
<b>anCol</b>	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
<b>abSetGet</b>	Bloco de código, opcional. Bloco de código no formato { u  if( Pcount( )>0, <var>:= u, <var> )} que o controle utiliza para atualizar a variável <var>. <var> deve ser tipo caracter ou numérica.
<b>aaItems</b>	Array de items caracteres, opcional. Lista de items selecionáveis.
<b>anWidth</b>	Numérico, opcional. Largura do controle em pixels.
<b>anHeight</b>	Numérico, opcional. Altura do controle em pixels.
<b>abChange</b>	Bloco de código, opcional. Executado quando o item selecionado é alterado.
<b>aoWnd</b>	Objeto, opcional. Janela ou controle onde o controle será criado.
<b>abValid</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.
<b>anClrFore</b>	Numérico, opcional. Cor de fundo do controle.
<b>anClrBack</b>	Numérico, opcional. Cor do texto do controle.
<b>alPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>IPar13</b>	Reservado.
<b>abLDBLClick</b>	Bloco de código, opcional. Executado quando acionado duplo click do botão esquerdo do mouse sobre o controle.
<b>aoFont</b>	Objeto, opcional. Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do controle.
<b>cPar16</b>	Reservado.
<b>IPar17</b>	Reservado.
<b>abWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

<b>aPar19</b>	Reservado.
<b>bPar20</b>	Reservado.
<b>IPar21</b>	Reservado.
<b>IPar22</b>	Reservado.
<b>abRightClick</b>	Bloco de código, opcional. Executado quando acionado click do botão direito do mouse sobre o controle.

**Métodos auxiliares:**

### Select

---

- Descrição:** Força a seleção de um item.
- Sintaxe:** Select( [anItem] )
- Parâmetros:**

nItem	Numérico, opcional. Posição do item a ser selecionado.
-------	--

- Retorno:**

Nenhum	-
--------	---

### Add

---

- Descrição:** Adiciona novo item.
- Sintaxe:** Add( cText, nPos )
- Parâmetros:**

cText	Caractere, obrigatório. Texto do item.
nPos	Numérico, obrigatório. Se 0 ou maior que o número de itens, insere o item no final da lista. Se valor entre 1 e número de itens, insere o item na posição informada, empurrando o item anterior para baixo.

- Retorno:**

Nenhum	-
--------	---

### Modify

---

- Descrição:** Modifica o texto de um item.
- Sintaxe:** Modify( cText, nPos )
- Parâmetros:**

cText	Caractere, obrigatório. Novo texto do item.
nPos	Numérico, obrigatório. Posição a ser modificada deve ser maior que 0 e menor ou igual que o número de itens.

- Retorno:**

Nenhum	-
--------	---

## **Del**

---

- Descrição:** Remove um item.
- Sintaxe:** Del( nPos )
- Parâmetros:**

nPos	Numérico, obrigatório. Posição a ser excluída, deve ser maior que 0 e menor ou igual que o número de itens.
------	---

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **Len**

---

- Descrição:** Retorna o número de itens.
- Sintaxe:** Len( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Numérico	Número de itens.
----------	------------------

## **Reset**

---

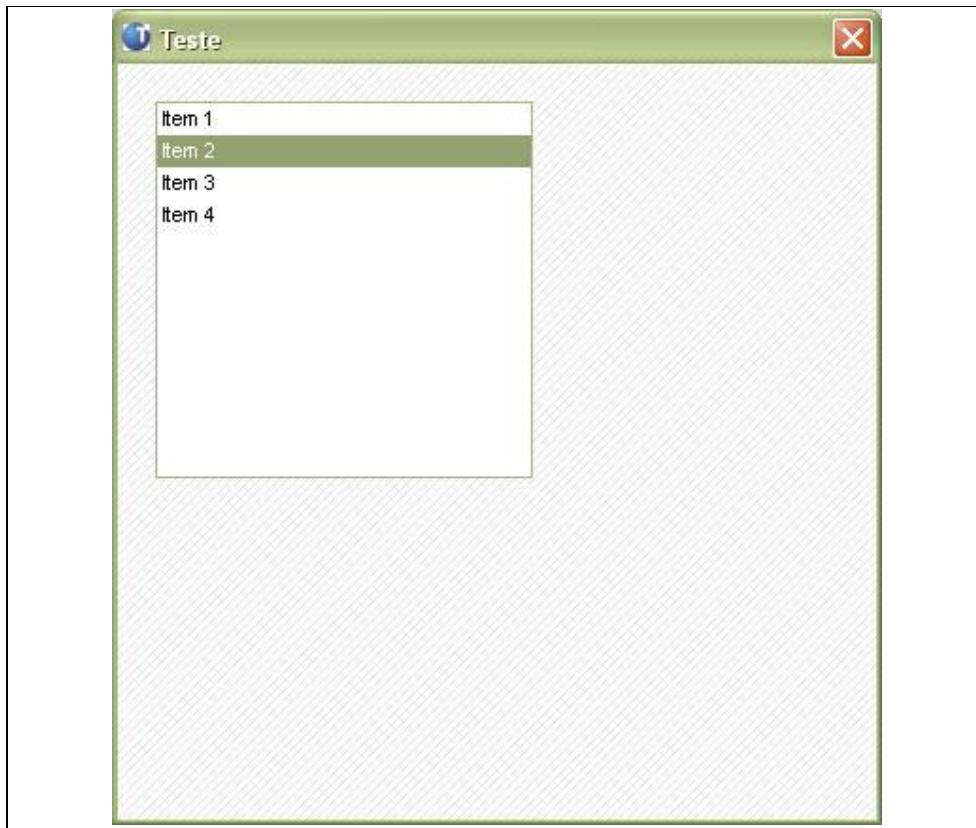
- Descrição:** Apaga todos os itens.
- Sintaxe:** Reset( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg, oList, nList:= 1
Local aItems:={}
Aadd(aItems,'Item 1')
Aadd(aItems,'Item 2')
Aadd(aItems,'Item 3')
Aadd(aItems,'Item 4')
DEFINE MSDIALOG oDlg FROM 0,0 TO 400,400 PIXEL TITLE 'Teste'
oList:= tListBox():New(10,10,{|u|if(Pcount()>0,nList:=u,nList)};
,aItems,100,100,,oDlg,,,.T.)
ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **TMENU()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Menu.

- Propriedades:**

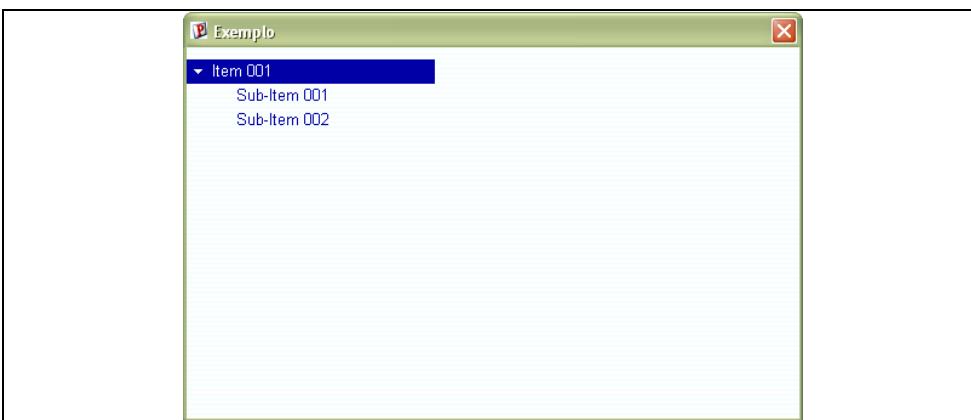
-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New ([nTop], [nLeft], [nHeight], [nWidth], [IPopUp], [cBmpName], [oWnd], [nClrNoSelect], [nClrSelect], [cArrowUpNoSel], [cArrowUpSel], [cArrowDownNoSel], [cArrowDownSel])

- Parâmetros:**

<b>nTop</b>	Numérico, opcional. Coordenada vertical em pixels.
<b>nLeft</b>	Numérico, opcional. Coordenada horizontal em pixels.
<b>nHeight</b>	Numérico, opcional. Altura do controle em pixels.
<b>nWidth</b>	Numérico, opcional. Largura do controle em pixels.
<b>IPopUp</b>	Lógico. Define se o objeto será um PoPup
<b>cBmpName</b>	Caracter opcional. Figura do Menu
<b>oWnd</b>	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
<b>nClrNoSelect</b>	Numerico opcional. Cor quando não selecionado
<b>nClrSelect</b>	Numerico opcional. Cor quando selecionado
<b>cArrowUpNoSel</b>	Caracter opcional, Define a figura da seta para cima quando não selecionado o item.
<b>cArrowUpSel</b>	Caracter opcional, Define a figura da seta para cima quando selecionado o item.
<b>cArrowDownNoSel</b>	Caracter opcional, Define a figura da seta para baixo quando não selecionado o item.
<b>cArrowDownSel</b>	Caracter opcional, Define a figura da seta para baixo quando selecionado o item.

- Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    oMenu := TMenu():New( 0,0,0,0,.F.,'',oDlg,CLR_WHITE,CLR_BLACK)
    // Adiciona Item ao Menu Principal
    oMenuItem1 := TMenuItem():New2( oMenu:Owner(),'Item 001','Item
001',,,)
        oMenu:Add( oMenuItem1 )

        // Adiciona sub-Itens
        oMenuItem2 := TMenuItem():New2( oMenu:Owner(),'Sub-Item
001',,,{||Alert('TMenuItem')})
        oMenuItem3 := TMenuItem():New2( oMenu:Owner(),'Sub-Item
002',,,{||Alert('TMenuItem')})
        oMenuItem1:Add( oMenuItem2 )
        oMenuItem1:Add( oMenuItem3 )

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

---

## **TMENUBAR()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Barra de Menu.

**Propriedades:**

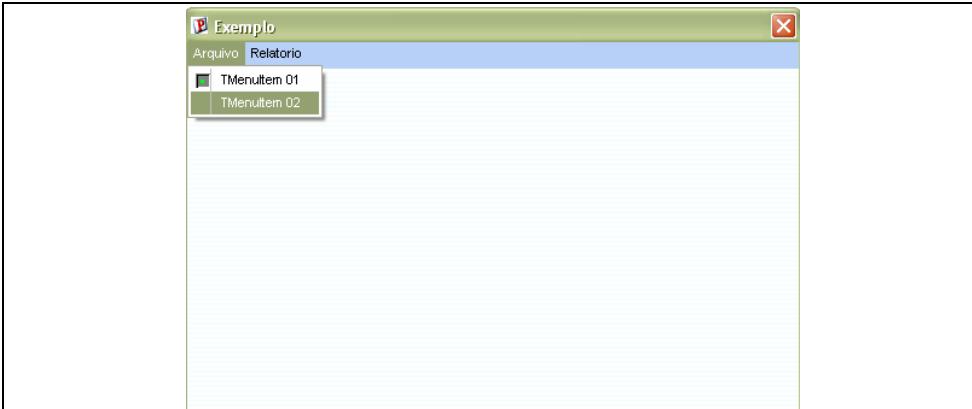
-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([oWnd])

**Parâmetros:**

oWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
------	--

**Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    // Monta um Menu Suspenso
    TMenuBar := TMenuBar():New(oDlg)
    TMenuBar:nClrPane := RGB(183,208,248) // Cor de fundo do Menu
    oMenu1 := TMenu():New(0,0,0,0,.T.,oDlg)
    oMenu2 := TMenu():New(0,0,0,0,.T.,oDlg)
    TMenuBar:AddItem('Arquivo' , oMenu1, .T.)
    TMenuBar:AddItem('Relatorio', oMenu2, .T.)

    // Cria Itens do Menu
    oMenuItem := TMenuItem():New(oDlg,'TMenuItem 01',,,,,
                                { ||Alert('TMenuItem 01')},,'AVGLBPAR1',,,,,,.T.)
    oMenu1:Add(oMenuItem)
    oMenu2:Add(oMenuItem)
    oMenuItem := TMenuItem():New(oDlg,'TMenuItem 02',,,,,
                                { ||Alert('TMenuItem 02')},,,,.T.)
    oMenu1:Add(oMenuItem)
    oMenu2:Add(oMenuItem)

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **TMETER()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tMeter, a qual exibe uma régua (gauge) de processamento, descrevendo o andamento de um processo através da exibição de uma barra horizontal.

**Propriedades:**

nTotal	Numérico. Número total de passos até o preenchimento da régua de processo.
lPercentage	Lógico. Se .T. considera o passo de movimentação em porcentagem.
nClrBar	Numérico. Cor da barra de andamento.

- Construtor:** New([anRow], [anCol], [abSetGet], [anTotal], [aoWnd], [anWidth], [anHeight], [lPar8], [alPixel], [oPar10], [cPar11], [alNoPerc], [anClrPane], [nPar14], [anClrBar], [nPar16], [lPar17])

**Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
abSetGet	Bloco de código, opcional. Bloco de código no formato {  u  if( Pcount( )>0, <var>:= u, <var> ) } que o controle utiliza para atualizar a variável <var>. <var> deve ser tipo numérico.
anTotal	Numérico, opcional. Numero total de passos até o preenchimento da régua de processo.

aoWnd	Objeto, opcional. Janela ou controle onde o controle sera criado.
anWidth	Numérico, opcional. Largura do controle em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
IPar8	Reservado.
alPixel	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
oPar10	Reservado.
cPar11	Reservado.
alNoPerc	Lógico, opcional. Se .T. (padrão) não considera os passos de atualização em porcentagem.
anClrPane	Numérico, opcional. Cor de fundo do controle.
nPar14	Reservado.
anClrBar	Numérico, opcional. Cor da barra de andamento.
nPar16	Reservado.
IPar17	Reservado.

**Métodos auxiliares:**

**Set**

---

- Descrição:** Atualiza a posição da régua de processamento.
- Sintaxe:** Set( [nVal] )
- Parâmetros:**

nVal	Numérico, opcional. Novo valor da posição da régua de processamento.
------	--

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
STATIC lRunning:=.F., lStop:=.F.
User Function Teste()
Local oDlg, oMeter, nMeter:=0, oBtn1, oBtn2
DEFINE MSDIALOG oDlg FROM 0,0 TO 400,400 TITLE 'Teste' Pixel
    oMeter:= tMeter():New(10,10,{|u|if(Pcount()>0,nMeter:=u,nMeter)}};
    ,100,oDlg,100,16,,.T.) // cria a régua
    // botão para ativar andamento da régua
    @ 30,10 BUTTON oBtn1 PROMPT 'Run' OF oDlg PIXEL ACTION RunMeter(oMeter)
    @ 50,10 BUTTON oBtn2 PROMPT 'Stop' OF oDlg PIXEL ACTION lStop:=.T.
ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
STATIC Function RunMeter(oMeter)
If lRunning
    Return
Endif
lRunning:= .T.

oMeter:Set(0)
// inicia a régua
While .T. .and. !lStop
    Sleep(1000) // pára 1 segundo
    ProcessMessages() // atualiza a pintura da janela, processa mensagens do windows
    nCurrent:= Eval(oMeter:bSetGet) // pega valor corrente da régua
    nCurrent+=10 // atualiza régua

    oMeter:Set(nCurrent)
    if nCurrent==oMeter:nTotal
        Return
    endif
Enddo

lRunning:= .F.
lStop:= .F.
Return
```

## **TMSGGRAPHIC()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Gráfico.

- Propriedades:**

I3D	Lógico, opcional Gráfico em 3D
IAxisVisib	Lógico, opcional Mostra eixos do gráfico

- Construtor:** New([anRow], [anCol], [aoWnd], [aoFont], [anClrText], [anClrBack], [anWidth], [anHeight])

- Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels.
anCol	Numérico, opcional. Coordenada horizontal em pixels.
aoWnd	Objeto, opcional. Janela ou controle onde a imagem deverá ser criada.
aoFont	Objeto, opcional. Fonte utilizada no gráfico.
anClrText	Caractere, Nome do resource.
anClrBack	Caractere, Nome do arquivo.
anWidth	Lógico, opcional. Exibe borda
anHeight	Objeto, opcional. Janela ou controle onde a imagem deverá ser criada.

- Métodos auxiliares:**

### **CreateSerie**

---

- Descrição:** Método para criação de uma serie para o gráfico.  
 **Sintaxe:** CreateSerie( [nSerieType], [cLegend], [nDecimals], [IShowValues] )  
 **Parâmetros:**

nSerieType	Numérico. Indica o tipo do gráfico GRP_LINE 1 GRP_AREA 2 GRP_POINT 3 GRP_BAR 4 GRP_PIE 10
cLegend	Caractere, opcional. Legenda da série.
nDecimals	Numérico, opcional. Numero de casas decimais dos valores.
IShowValues	Lógico, opcional. Mostra valores

- Retorno:**

Numérico	Numero da série criada.
----------	-------------------------

## Add

---

- Descrição:** Método para adicionar um item ao gráfico.
- Sintaxe:** Add(nSerie, nVal, cLegend, nColor )
- Parâmetros:**

nSerie	Numérico. Serie a ser inserido o item.
nVal	Numérico. Valor do item.
cLegend	Caractere. Legenda do item.
nColor	Numérico, Cor do item.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## SetGradient

---

- Descrição:** Método para setar o fundo do gráfico com uma cor gradiente em determinada direção.
- Sintaxe:** SetGradient(nDirection, StartColor, EndColor )
- Parâmetros:**

nDirection	Numérico. Indica direção do gradiente. GDTOPBOTTOM 1 GDBOTTOMTOP 2 GDLEFTRIGHT 3 GDRIGHTLEFT 4
StartColor	Numérico. Cor inicial.
EndColor	Numérico. Cor final.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## SetTitle

---

- Descrição:** Define o titulo do gráfico.
- Sintaxe:** SetTitle(cTitle, cTitle2, Color, Alignmet, lFoot)
- Parâmetros:**

cTitle	Caractere. Primeiro titulo.
cTitle2	Caractere. Segundo titulo.
Color	Numérico. Cor do titulo.
Alignmet	Numérico. Alinhamento do titulo. A_LEFTJUST 1 A_RIGHTJUS 2 A_CENTER 3
lFoot	Lógico. Indica titulo no rodapé do gráfico.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SetLegenProp**

---

- Descrição:** Método para setar propriedades da legenda.
- Sintaxe:** SetLegenProp(Align, Color, Style, Visible )
- Parâmetros:**

Align	Numerico. Alinhamento da legenda. GRP_SCRTOP 1 GRP_SCRLEFT 2 GRP_SCRBOTTOM 3 GRP_SCRRIGHT 4
Color	Numérico. Cor da legenda.
Style	Numérico. Estilo da legenda. GRP_AUTO 1 GRP_SERIES 2 GRP_VALUES 3 GRPLASTVAL 4
Visible	Lógico. Indica se o título será visível.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SetMargins**

---

- Descrição:** Método para setar as margens superior, inferior e laterais do gráfico.
- Sintaxe:** SetMargins(nTop, nLeft, nBottom, nRight )
- Parâmetros:**

nTop	Numérico. Posição em relação ao topo do gráfico.
nLeft	Numérico. Posição em relação a esquerda.
nBottom	Numérico. Posição em relação ao rodapé.
nRight	Numérico. Posição em relação a direita.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SetRangeY**

---

- Descrição: Define a escala dos valores do eixo Y**
- Sintaxe:** SetRangeY (min, max, delta).
- Parâmetros:**

Min	valor inicial
Max	valor final
delta	intervalo entre os valores [ opcional, calculado automaticamente ]

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SaveToBMP**

---

- Descrição:** Método para salvar o gráfico atual em um bitmap no server(**Obsoleto, ver SaveToImage**).
- Sintaxe:** SaveToBMP(BmpName, PathToWrite )
- Parâmetros:**

BmpName	Caractere, Nome do da figura a ser salva.
PathToWrite	Caractere. Path no Server onde será salva a figura.

- Retorno:**

Lógico	Indica se a imagem foi salva corretamente.
--------	--

## **SaveToImage**

---

- Descrição:** Método para salvar o gráfico atual em um formato de pré-determinado no server.
- Sintaxe:** SaveToImage(BmpName, PathToWrite, TypeImage)
- Parâmetros:**

BmpName	Caractere, Nome do da figura a ser salva.
PathToWrite	Caractere. Path no Server onde será salva a figura.
TypeImage	Caractere. Tipo da Figura (Default 'JPEG') Tipos suportados: JPEG, PNG, BMP

- Retorno:**

Lógico	Indica se a imagem foi salva corretamente.
--------	--

## **DelSerie**

---

- Descrição:** Método para deletar uma série do grafico.
- Sintaxe:** DelSerie ( nSerie )
- Parâmetros:**

nSerie	Numérico. Serie a ser deletada.
--------	---------------------------------

- Retorno:**

Lógico	Indica se a série foi removida do gráfico
--------	---

## **ZoomIn**

---

- Descrição:** Método para efetuar zoom interno ( + ).
- Sintaxe:** ZoomIn()
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nenhum	-
--------	---

## **ZoomOut**

---

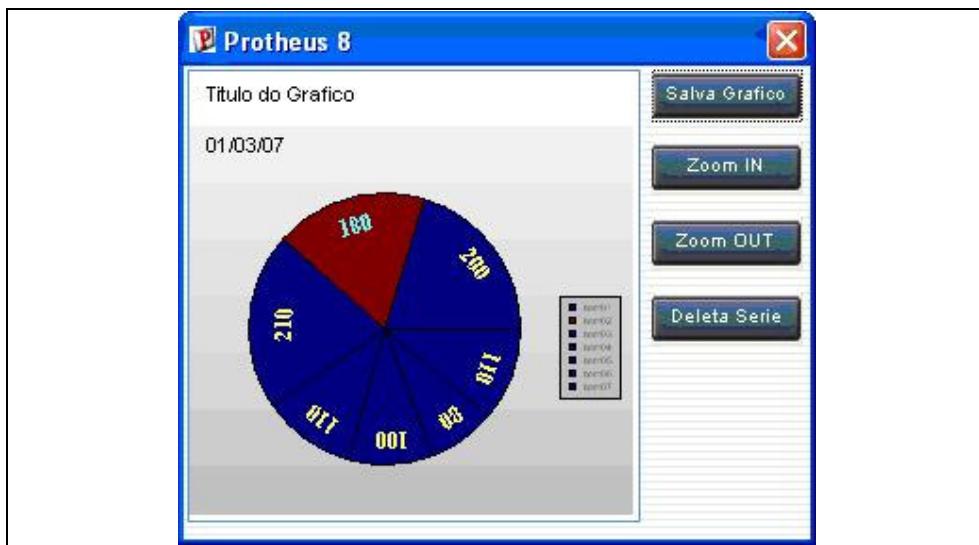
- Descrição:** Método para efetuar zoom externo ( - ).
- Sintaxe:** ZoomOut()
- Parâmetros:**

**Nenhum** | -

- Retorno:**

**Nenhum** | -

- Aparência:**



*Anotações* \_\_\_\_\_

---

---

---

---

## **Exemplo:**

---

```
#include 'MSGGRAPHI.CH'
User Function tMsGraphicTst()

    DEFINE MSDIALOG oDlg TITLE '' FROM 0,0 TO 250,330 PIXEL

        @ 001, 001 MSGGRAPHIC oGraphic SIZE 120,120 OF oDlg

        oGraphic:SetTitle('Titulo do Grafico', DTOC(Date()), CLR_BLACK,
A_LEFTJUST, GRP_TITLE )
        oGraphic:SetMargins(2,6,6,6)
        oGraphic:SetGradient(GDBOTTOMTOP, CLR_HGRAY, CLR_WHITE)
        oGraphic:SetLegenProp(GRP_SCRRIGHT, CLR_HGRAY, GRP_AUTO,.T.)

        nSerie      :=   oGraphic>CreateSerie( 10 )
        oGraphic:l3D := .T. // Grafico em 3D
        oGraphic:lAxisVisib     := .T.      // Mostra os eixos

        // Itens do Grafico
        oGraphic:Add(nSerie, 200, 'Item01', CLR_BLUE )
        oGraphic:Add(nSerie, 180, 'Item02', CLR_RED  )
        oGraphic:Add(nSerie, 210, 'Item03', CLR_BLUE )
        oGraphic:Add(nSerie, 110, 'Item04', CLR_BLUE )
        oGraphic:Add(nSerie, 100, 'Item05', CLR_BLUE )
        oGraphic:Add(nSerie, 080, 'Item06', CLR_BLUE )
        oGraphic:Add(nSerie, 110, 'Item07', CLR_BLUE )

        @ 001, 124 BUTTON 'Salva Grafico' SIZE 40,14 OF oDlg PIXEL ACTION
        oGraphic:SaveToBMP('Exemplo.bmp','\web\'')
            @ 020, 124 BUTTON 'Zoom IN'           SIZE 40,14 OF oDlg PIXEL ACTION
        oGraphic:ZoomIn()
            @ 040, 124 BUTTON 'Zoom OUT'         SIZE 40,14 OF oDlg PIXEL ACTION
        oGraphic:ZoomOut()
            @ 060, 124 BUTTON 'Delete Serie'    SIZE 40,14 OF oDlg PIXEL ACTION
        oGraphic:DelSerie(nSerie)

        ACTIVATE MSDIALOG oDlg CENTERED

Return
```



**Anotações**

---

---

---

---

## **TMSGBAR()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Rodapé.

- Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New ([oWnd], [cPrompt], [ICentered], [IClock], [IDate], [IKbd], [nClrFore], [nClrBack], [oFont], [Inset], [imageName])

- Parâmetros:**

oWnd	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
cPrompt	Caracter. Descrição na Barra
ICentered	Logico. Define centralização do texto
IClock	Nao utilizado
IDate	Nao utilizado
IKbd	Nao utilizado
nClrFore	Numerico, opcional. Define cor da fonte da barra
nClrBack	Numerico, opcional. Define cor do fundo da barra
oFont	Objeto, opcional. Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.
Inset	Nao utilizado
imageName	Caracter, opcional. Insere figura lateral esquerda

- Métodos auxiliares:**

### **AddItem**

---

- Descrição:** Método insere um subitem no rodapé

- Sintaxe:** AddItem( oTMsgItem )

- Parâmetros:**

oTMsgItem	Objeto do tipo TMsgItem que será inserido como subitem do rodapé.
-----------	---

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **SetMsg**

---

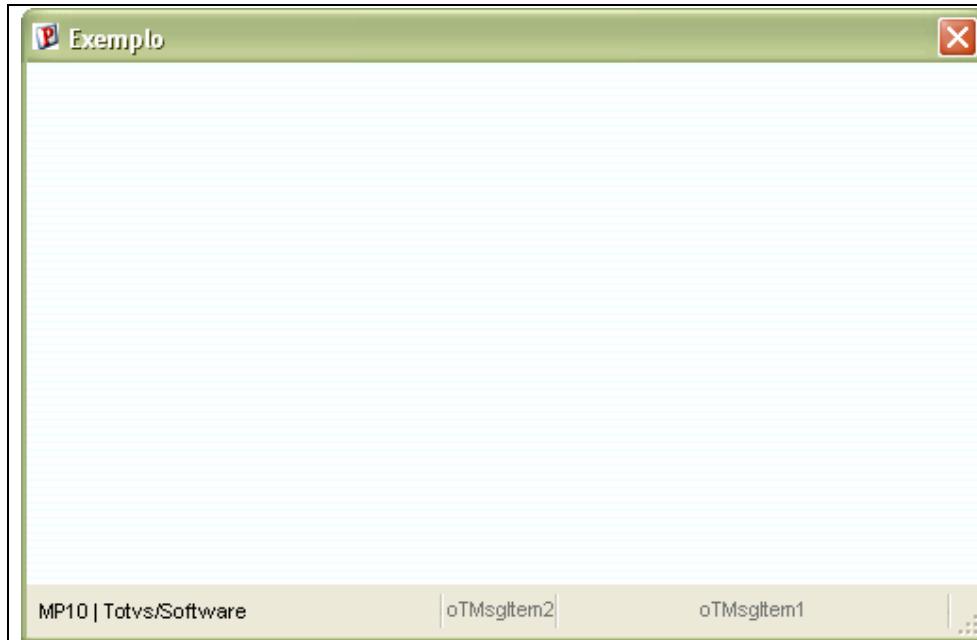
- Descrição:** Define a descrição da barra de rodapé
- Sintaxe:** SetMsg(cTexto )
- Parâmetros:**

cTexto	Texto a ser inserido na barra
--------	-------------------------------

- Retorno:**

Nenhum	-
--------	---

- Aparência:**



### **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    oMsgBar := TMsgBar():New(oDlg, 'MP10 | Totvs/Software',
    .F.,.F.,.F.,.F., RGB(116,116,116),,.F.)
    oMsgItem1 := TMsgItem():New( oMsgBar,'oMsgItem1', 204,,,.T.,
    {||})
    oMsgItem2 := TMsgItem():New( oMsgBar,'oMsgItem2', 040,,,.T.,
    {||}Alert('Item da Barra Acionado')) }

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## TMSGITEM()

- Descrição:** Classe de objetos visuais do tipo controle – utilizado para adicionar mensagens em outros objetos, como barras e menus.

**Propriedades:**

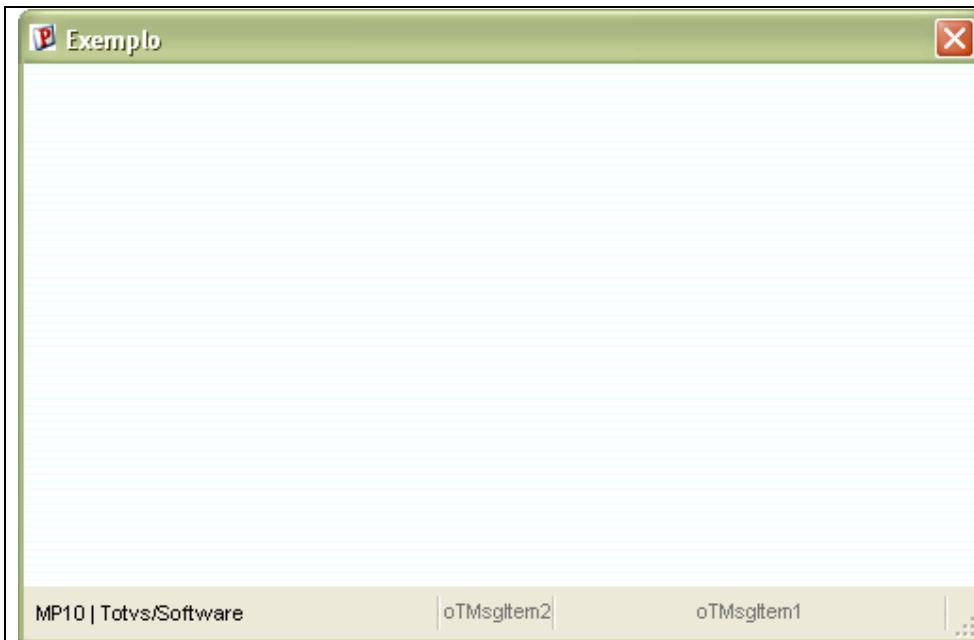
-	Herdadas das classes superiores
---	---------------------------------

- Construtor:** New( <oMsgBar>, <cMsg>, <nSize>, <oFont>, <nClrFore>, <nClrBack>, IEnable , [<{bAction}>], cImagen )

**Parâmetros:**

<b>oMsgBar</b>	Objeto do tipo barra ou menu no qual será adicionado o item de mensagem.
<b>cMsg</b>	Mensagem que será exibida pelo objeto
<b>nSize</b>	Tamanho da fonte do texto
<b>oFont</b>	Fonte do texto
<b>nClrFore</b>	Cor da fonte do texto
<b>nClrBack</b>	Cor do fundo do objeto
<b>IEnable</b>	Indica se o objeto está habilitado.
<b>bAction</b>	Bloco de código executado quando o objeto é ativado
<b>cImagen</b>	Imagen a ser vinculada no objeto.

**Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'
    oMsgBar := TMsgBar():New(oDlg, 'MP10 | Totvs/Software',
.F.,.F.,.F., RGB(116,116,116),,.F.)
    oMsgItem1 := TMsgItem():New( oMsgBar,'oMsgItem1', 204,,,.T.,
{||})
    oMsgItem2 := TMsgItem():New( oMsgBar,'oMsgItem2', 040,,,.T.,
{||}Alert('Item da Barra Acionado')) )

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

---

## **TMULTIBTN()**

---

- Descrição:** Classe de objetos visuais do tipo controle - Múltiplos botões.

- Propriedades:**

bAction	Bloco de código. Executado ao precionar o botão esquerdo do mouse.
---------	--

- Construtor:** New([nRow], [nCol], [cTitle], [oWnd], [bAction], [nWidth], [nHeight], [imgName], [ori], [cMsg], [btnPerLine])

- Parâmetros:**

nRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
nCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres
oWnd	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
bAction	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for pressionado.
nWidth	Numérico, opcional. Largura em pixels
nHeight	Numérico, opcional. Altura em pixels.
imgName	Caracter. Imagem a ser inserida ao lado esquerdo do componente
ori	Numerico, opcional. Orientação dos botões.
cMsg	Caracter, Opicional. Descrição do tipo tooltip(hint) do item
btnPerLine	Numerico, opcional. Quantidade de botões por linha

**Métodos auxiliares:**

**LoadImage**

- Descrição:** Muda a figura a esquerda do componente
- Sintaxe:** LoadImage(cImagen)
- Parâmetros:**

cImagen	Nome da imagem que será carregada pelo objeto
---------	---

- Retorno:**

Nenhum	-
--------	---

**AddButton**

- Descrição:** Insere um botão
- Sintaxe:** AddButton(cTexto)
- Parâmetros:**

cTexto	Texto para exibição no botão inserido.
--------	--

- Retorno:**

Nenhum	-
--------	---

**SetTitle**

- Descrição:** Muda o titulo e o numero de botões por linha
- Sintaxe:** SetTitle(cTexto, nBotoes)
- Parâmetros:**

cTexto	Texto que será atribuído aos botões
nBotoes	Número de botões por linha

- Retorno:**

Nenhum	-
--------	---

**SetFont**

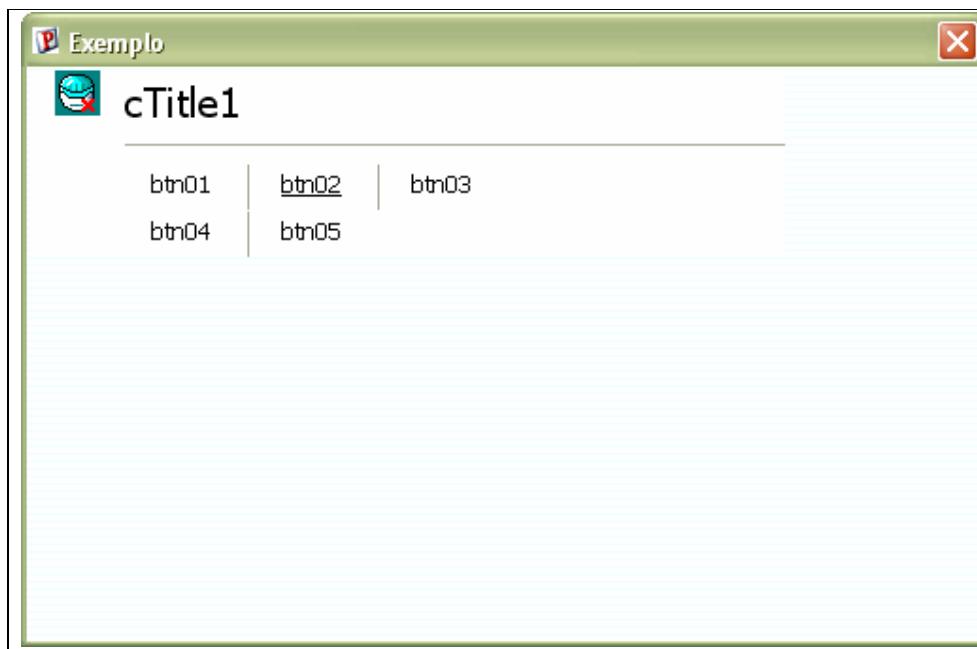
- Descrição:** Muda a fonte do título e dos botões
- Sintaxe:** SetTitle(cFontTit, nSizeTit, nFontBut, nSizeBut)
- Parâmetros:**

cFontTit	Nome da fonte que será atribuída ao título
nSizeTit	Tamanho da fonte do título
nFontBut	Nome da fonte que será atribuída aos botões
nSizeBut	Tamanho da fonte dos botões

- Retorno:**

Nenhum	-
--------	---

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'

    otMultiBtn := tMultiBtn():New( 01,01,'cTitle1',oDlg,;
                                { |x|Alert(Str(x)) },200,150,'afastamento',0,'cTitle',3
)

    otMultiBtn:SetFonts('Tahoma',16,'Tahoma',10)
    otMultiBtn:AddButton('btn01')
    otMultiBtn:AddButton('btn02')
    otMultiBtn:AddButton('btn04')
    otMultiBtn:AddButton('btn05')

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
.
```

## **TMULTIGET()**

---

- Descrição:** Classe de objetos visuais do tipo controle - edição de texto de múltiplas linhas.

- Propriedades:**

IWordWrap	Lógico. Se .T., faz quebra automática de linhas.
-----------	--

- Construtor:** New([anRow], [anCol], [abSetGet], [aoWnd], [anWidth], [anHeight], [aoFont], [alHScroll], [anClrFore], [anClrBack], [oPar11], [alPixel], [cPar13], [IPar14], [abWhen], [IPar16], [IPar17], [alReadOnly], [abValid], [bPar20], [IPar21], [alNoBorder], [alNoVScroll])

- Parâmetros:**

Parâmetro	
anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
abSetGet	Bloco de código, opcional. Bloco de código no formato {  u  if( Pcount( )>0, <var>:= u, <var> ) } que o controle utiliza para atualizar a variável <var>. <var> deve ser tipo caracter.
aoWnd	Objeto, opcional. Janela ou controle onde o controle será criado.
anWidth	Numérico, opcional. Largura do controle em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
aoFont	Objeto, opcional. Objeto tipo tFont utilizado para definir as características da fonte utilizada para exibir o conteúdo do controle.
alHScroll	Lógico, opcional. Se .T., habilita barra de rolagem horizontal.
anClrFore	Numérico, opcional. Cor de fundo do controle.
anClrBack	Numérico, opcional. Cor do texto do controle.
oPar11	Reservado.
alPixel	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
cPar13	Reservado.
IPar14	Reservado.
abWhen	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
IPar16	Reservado.
IPar17	Reservado.
alReadOnly	Lógico, opcional. Se .T. o controle só permitira leitura.
abValid	Bloco de código, opcional. Executado quando o conteúdo do controle deve ser validado, deve retornar .T. se o conteúdo for válido e .F. quando o conteúdo for inválido.
bPar20	Reservado.
IPar21	Reservado.

alNoBorder	Lógico, opcional. Se .T. cria controle sem borda.
alNoVScroll	Lógico, opcional. Se .T., habilita barra de rolagem vertical.

**Métodos auxiliares:**

**EnableVScroll**

- Descrição:** Habilita a barra de rolagem vertical.
- Sintaxe:** EnableVScroll( IEnable )
- Parâmetros:**

IEnable	Lógico, obrigatório. Se .T. habilita se .F. desabilita a barra de rolagem.
---------	--

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**EnableHScroll**

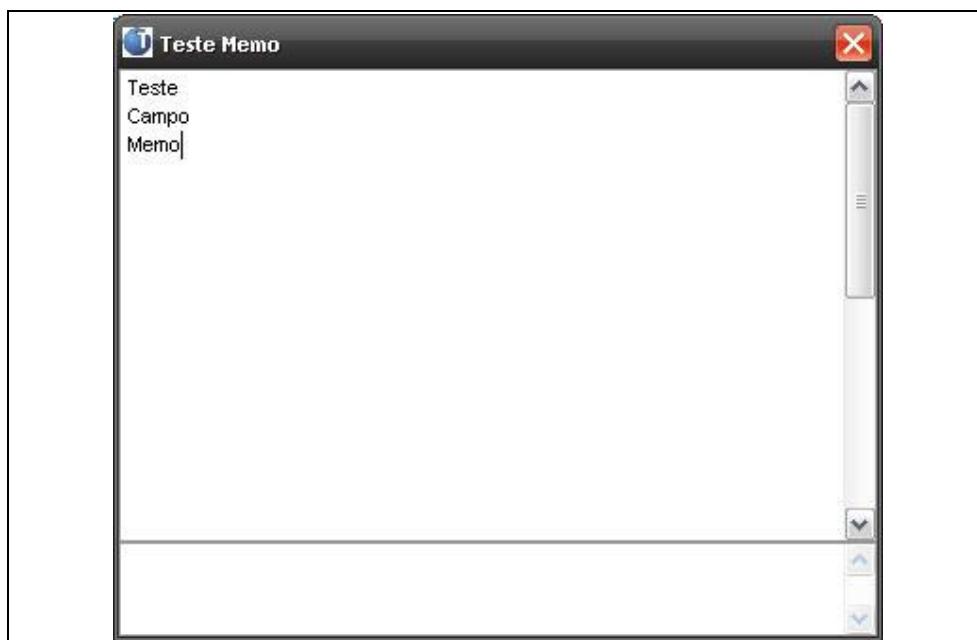
- Descrição:** Habilita a barra de rolagem horizontal.
- Sintaxe:** EnableHScroll( IEnable )
- Parâmetros:**

IEnable	Lógico, obrigatório. Se .T. habilita se .F. desabilita a barra de rolagem.
---------	--

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Aparência:**



### **Exemplo:**

---

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

Local oDlg, oMemo, cMemo:= space(50)

DEFINE MSDIALOG oDlg FROM 0,0 TO 400,400 PIXEL TITLE "My test"

oMemo:= tMultiget():New(10,10,{|u|if(Pcount()>0,cMemo:=u,cMemo)};
,oDlg,100,100,,,.T.)

@ 200,10 BUTTON oBtn PROMPT "Fecha" OF oDlg PIXEL ACTION oDlg:End()

ACTIVATE MSDIALOG oDlg CENTERED

MsgStop(cMemo)

Return NIL
```

---

### **TOLECONTAINER()**

---

- Descrição:** Classe de objetos visuais do tipo controle, a qual permite a criação de um botão vinculado a um objeto OLE.

- Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([nRow], [nCol], [nWidth], [nHeight], [oWnd], [IAutoActivate], [cFileName])

- Parâmetros:**

nRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
nCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
nWidth	Numérico, opcional. Largura do botão em pixels
nHeight	Numérico, opcional. Altura do botão em pixels.
oWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
IAutoActivate	Nao utilizado
cFileName	Caracter. Endereço do arquivo Ole a ser aberto

**Métodos auxiliares:**

**OpenFromFile**

---

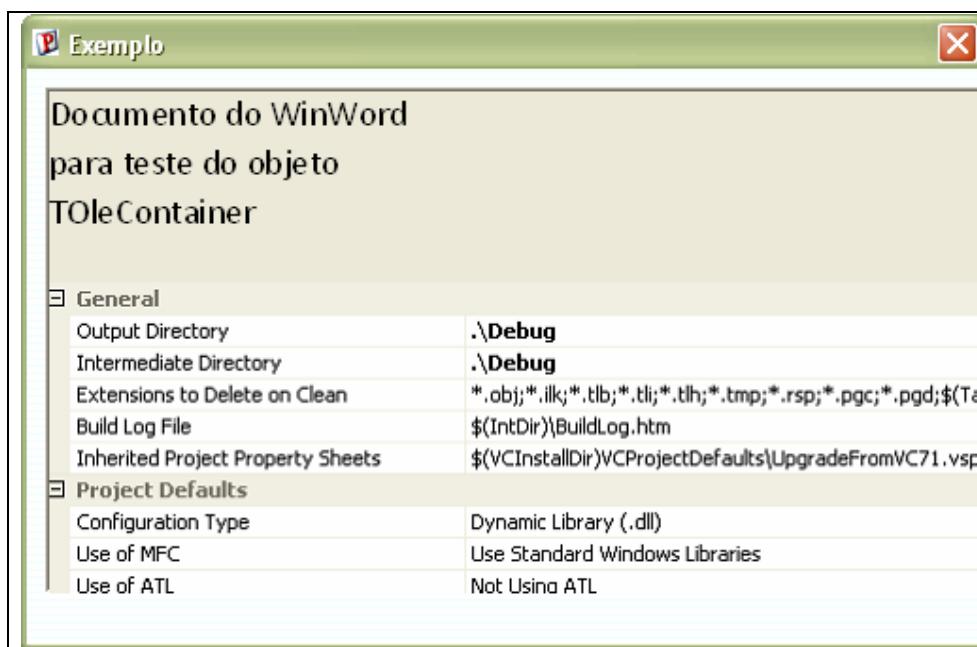
- Descrição:** Abre arquivo do tipo OLE.
- Sintaxe:** New([cFileName, [IAsIcon], [allowInPlace]])
- Parâmetros:**

cFileName	Caracter. Endereço do arquivo Ole a ser aberto
IAsIcon	Logico. Define objeto Ole como Icone
allowInPlace	Logico opcional. Permite abertura de arquivo local

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Aparência:**



**Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'

    // TOleContainer
    oTOleContainer := TOleContainer():New(
05,05,306,134,oDlg,.T.,'C:\Lixo\TesteRemote.doc' )

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **TPAGEVIEW()**

---

- Descrição:** Classe de objetos visuais do tipo controle, que permite a visualização de arquivos no formato gerado pelo spool de impressão do Protheus.

**Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([nCol], [nRow], [nWidth], [nHeight], [oPrinter], [oWnd], [nPageWidth], [nPageHeight])

**Parâmetros:**

nCol	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
nRow	Numérico, opcional. Coordenada horizontal em pixels ou caracteres
nWidth	Numérico, opcional. Largura em pixels
nHeigth	Numérico, opcional. Altura em pixels.
oPrinter	Objeto do tipo TMsPrinter()
oWnd	Objeto, opcional. Janela ou controle onde a botão deverá ser criado
nPageWidth	Numérico, opcional. Largura em pixels da pagina
nPageHeight	Numérico, opcional. Altura em pixels da pagina

**Métodos auxiliares:**

### **PageSize**

---

- Descrição:** Define o tamanho da pagina.  
 **Sintaxe:** AddItem(nLargura, nAltura)  
 **Parâmetros:**

nLargura	Lagura da página
nAltura	Altura da página

- Retorno:**

<b>Nenhum</b>	-
---------------	---

### **PrintPage**

---

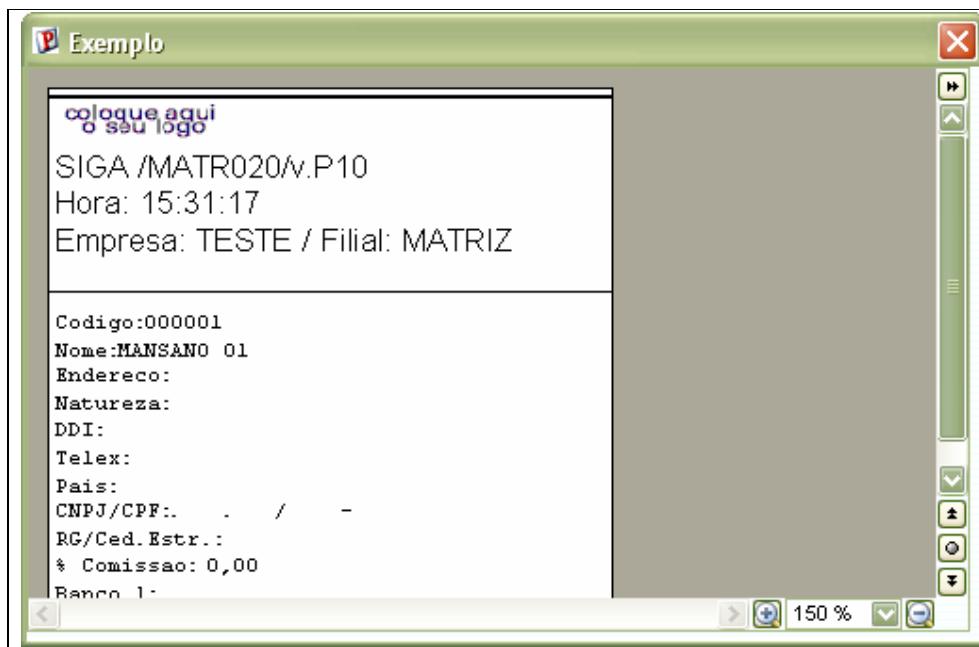
- Descrição:** Imprime uma determinada página.  
 **Sintaxe:** PrintPage(nNrPagina)  
 **Parâmetros:**

nNrPagina	Página que será impressa
-----------	--------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'

    // TPageView
    __RelDir := WSPLRelDir()
    oPrinter := TMSPrinter():New()
    oPrinter:SetFile(__RELDIR + 'matr020.prt',.F.)
    oTPageView := TPageView():New(
0,0,500,300,oPrinter,oTFolder2:aDialogs[07],550,350 )
    oTPageView:Reset(400,400)
    oTPageView:PrintPage(1)
    oTPageView>Show()
    oTPageView:nZoom := 150

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## **TPANEL()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tPanel, a qual permite criar um painel estático, onde podem ser criados outros controles com o objetivo de organizar ou agrupar componentes visuais.

- Propriedades:**

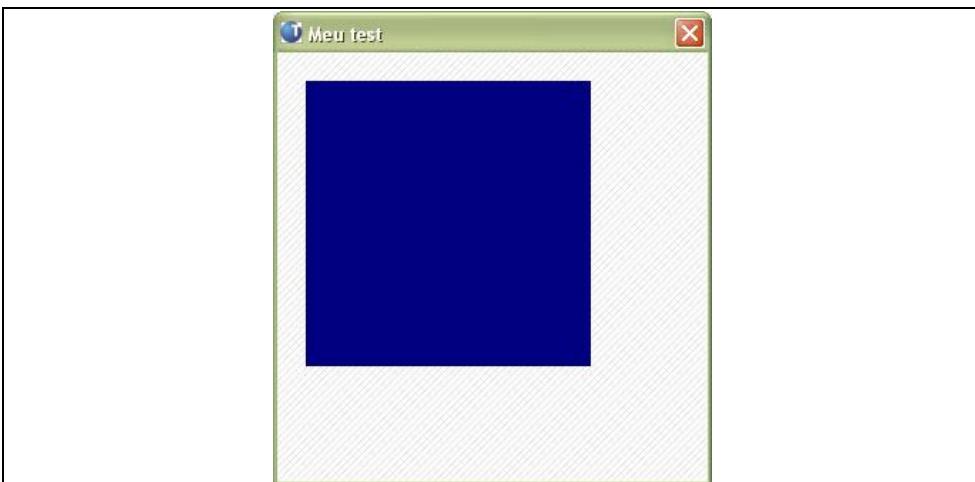
-	Herdadas das classes superiores
---	---------------------------------

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

- Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels.
anCol	Numérico, opcional. Coordenada horizontal em pixels.
acText	Caractere, opcional. Texto a ser exibido ao fundo.
aoWnd	Objeto, opcional. Janela ou controle onde será criado o objeto.
alCentered	Lógico, opcional. Se .T. exibe o texto de título ao centro do controle.
IPar6	Reservado.
anClrText	Numérico, opcional. Cor do texto do controle.
anClrBack	Numérico, opcional. Cor do fundo do controle.
anWidth	Numérico, opcional. Largura do controle em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
alLowered	Lógico, opcional. Se .T. exibe o painel rebaixado em relação ao controle de fundo.
alRaised	Lógico, opcional. Se .T. exibe a borda do controle rebaixada em relação ao controle de fundo.

- Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg, oButton, oCombo, cCombo, cGet1:='Teste'
DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu test'
  oPanel:= tPanel():New(10,10,"",oDlg,,,,CLR_BLUE,100,100) // cria o
  painel
  @ 10,10 BUTTON oBtn1 PROMPT 'hide' OF oPanel ACTION oPanel:Hide() // cria
  botão sobre o painel
  @ 200,10 BUTTON oBtn2 PROMPT 'show' OF oDlg ACTION oPanel>Show() // cria
  botão fora o painel
ACTIVATE MSDIALOG oDlg CENTERED
Return
```

## **TRADMENU()**

---

- Descrição:** Classe de objetos visuais do tipo controle – TRadMenu, a qual permite criar um controle visual no formato Radio Button.

**Propriedades:**

<b>bChange</b>	Bloco de código disparado na mudança de item selecionado
<b>bSetGet</b>	Bloco de código disparado na mudança de item selecionado, responsável pela mudança de valor da variável numérica que indica o item selecionado.
<b>bWhen</b>	Bloco de código que permite ou não a alteração do objeto
<b>bValid</b>	Bloco de código executado na saída do objeto.

- Construtor:** New([nRow], [nCol], [aItems], [bSetGet], [oDlg], [aHelpIds], [bChange], [nClrText], [nClrPane], [cMsg], [lUpdate], [bWhen], [nWidth], [nHeight], [bValid], [IDesign], [l3d], [IPixel])

**Parâmetros:**

nRow	Numérico, opcional. Coordenada vertical
nCol	Numérico, opcional. Coordenada horizontal
aItems	Vetor, elementos do Radio
bSetGet	Code-block, Responsável pela setagem de valor
oDlg	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
aHelpIds	Não utilizado
bChange	Bloco de código, opcional. Executado quando o item selecionado é alterado.
nClrText	Numérico, opcional. Cor do texto da janela.
nClrPane	Numérico, opcional. Cor de fundo da janela.
cMsg	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
lUpdate	Não utilizado
bWhen	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o

	controle deve permanecer habilitado ou .F. se não.
nWidth	Numérico, opcional. Largura do objeto
nHeight	Numérico, opcional. Altura do objeto
bValid	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.
IDesign	Não utilizado
I3d	Não utilizado
IPixel	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.

**Métodos auxiliares:**

**SetOption**

- Descrição:** Seleciona um item.
- Sintaxe:** SetOption(**nItem**)
- Parâmetros:**

<b>nItem</b>	Item que será selecionado
--------------	---------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**EnableItem**

- Descrição:** Habilita/Desabilita um Item
- Sintaxe:** EnableItem(**nItem**, **IStatus**)
- Parâmetros:**

<b>nItem</b>	Item que será modificado
<b>IStatus</b>	Status que será atribuído ( .T. – Enable , .F. – Disable)

- Retorno:**

<b>Nenhum</b>	-
---------------	---

**Enable**

- Descrição:** Habilita um item.
- Sintaxe:** Enable(**nItem**)
- Parâmetros:**

<b>nItem</b>	Item que será habilitado
--------------	--------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## Disable

---

- Descrição:** Desabilita um item
- Sintaxe:** Disable(nItem)
- Parâmetros:**

<b>nItem</b>	Item que será desabilitado
--------------	----------------------------

- Retorno:**

<b>Nenhum</b>	-
---------------	---

- Aparência:**



*Anotações*

---

---

---

---

---

## **Exemplo:**

---

```
#include 'protheus.ch'
user function TRadMenu()
    DEFINE MSDIALOG oDlg FROM 0,0 TO 270,400 PIXEL TITLE 'Exemplo da
TCBrowse'
        // Variavel numerica que guarda o item selecionado do Radio
        nRadio := 1
        // Cria o Objeto
        oRadio := TRadMenu():New
(01,01,['Item01','Item02','Item03','Item04','Item05'],;

, oDlg,,,,,,200,200,,,,,T.)
        // Seta Eventos
        oRadio:bchange     := {|| Alert('bChange') }
        oRadio:bSetGet     := {|u|Iif (PCount() == 0, nRadio, nRadio:=u) }
        oRadio:bWhen       := {|| .T. }
        oRadio:bValid      := {|| Alert('bValid') }
        // Principais comandos
        oBtn   := TButton():New( 060, 001, 'Retorna item selecionado',;
        oDlg,{|| Alert(nRadio)           }, 120,
010,,,F.,,T.,,F.,,F.,,,F. )
        oBtn   := TButton():New( 070, 001, 'SetOption(2) (Seta um
item)',;
        oDlg,{|| oRadio:SetOption(2)      }, 120,
010,,,F.,,T.,,F.,,F.,,,F. )
        oBtn   := TButton():New( 080, 001, 'EnableItem(2,.T.) (Habilita
item)',;
        oDlg,{|| oRadio:EnableItem(2,.T.) }, 120,
010,,,F.,,T.,,F.,,F.,,,F. )
        oBtn   := TButton():New( 090, 001, 'EnableItem(2,.F.) (Desabilita
item)',;
        oDlg,{|| oRadio:EnableItem(2,.F.) }, 120,
010,,,F.,,T.,,F.,,F.,,,F. )
        oBtn   := TButton():New( 100, 001, 'Enable(3) (Habilita
item)',;
        oDlg,{|| oRadio:Enable(3)         }, 120,
010,,,F.,,T.,,F.,,F.,,,F. )
        oBtn   := TButton():New( 110, 001, 'Disable(3) (Desabilita
item)',;
        oDlg,{|| oRadio:Disable(3)       }, 120,
010,,,F.,,T.,,F.,,F.,,,F. )

        ACTIVATE MSDIALOG oDlg CENTERED
Return
```

## **TSBROWSE()**

---

- Descrição:** Classe de objetos visuais do tipo controle – TSBrowse, a qual permite criar um controle visual do tipo Grid.

**Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>nAt</b>	Linha atualmente selecionada / posicionada no objeto
<b>nLen</b>	Número total de linhas do objeto

- Construtor:** New([nRow], [nCol], [nWidth], [nHeight], [oWnd], [bChange], [nHWidth], [oFont], [nLines])

**Parâmetros:**

nRow	Numérico, opcional. Coordenada vertical
nCol	Numérico, opcional. Coordenada horizontal
nWidth	Numérico, opcional. Largura do objeto
nHeight	Numérico, opcional. Altura do objeto
oWnd	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
bChange	Bloco de código, na mudança de linha
nHWidth	Não utilizado
oFont	Objeto, opcional. Fonte
nLines	Númerico. Nr de linhas por célula

**Métodos auxiliares:**

### **GoUp**

---

- Descrição:** Salta uma linha para cima.  
 **Sintaxe:** GoUp( )  
 **Parâmetros:**

Nenhum | -

- Retorno:**

Nil

### **GoDown**

---

- Descrição:** Salta uma linha para baixo.  
 **Sintaxe:** GoDown( )  
 **Parâmetros:**

Nenhum | -

- Retorno:**

Nil

## **GoTop**

---

- Descrição:** Salta para primeira linha.
- Sintaxe:** GoTop( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

## **GoBottom**

---

- Descrição:** Salta para ultima linha.
- Sintaxe:** GoBottom( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

## **RowCount**

---

- Descrição:** Retorna numero de linhas visiveis.
- Sintaxe:** RowCount( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

## **LEditCell**

---

- Descrição:** Edita o valor de uma coluna.
- Sintaxe:** LEditCell( aList, oList, cPicture, nCol )
- Parâmetros:**

aList	Vetor da Browse onde estão os valores da mesma
oList	Objeto, Browse a ser editado
cPicture	Caracter, picture necessária para edição do campo
nCol	Numérico, coluna a ser editada.

- Retorno:**

Nil
-----

**Aparência:**



*Anotações*

---

---

---

---

## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 302,402 PIXEL TITLE 'Exemplo'

    aBrowse := {{'CLIENTE 001','RUA CLIENTE 001','BAIRRO CLIENTE
001'},;
                {'CLIENTE 001','RUA CLIENTE 001','BAIRRO CLIENTE
001'},;
                {'CLIENTE 001','RUA CLIENTE 001','BAIRRO CLIENTE 001'}
}
    oBrowse := TSBrowse():New(01,01,245,150,oDlg,,16,,5)
    nForeCor := CLR_GRAY
    nBackCor := CLR_WHITE
    oBrowse:addColumn( TcColumn():New('Nome',,,{|| nForeCor },{|| 
nBackCor }) )
    oBrowse:addColumn( TcColumn():New('Endereço',,,{|| nForeCor },{|| 
nBackCor }) )
    oBrowse:addColumn( TcColumn():New('Bairro',,,{|| nForeCor },{|| 
nBackCor }) )
    oBrowse:SetArray(aBrowse)

    // Principais commandos
    TButton():New(160,001,'GoUp()',oDlg,{|| oBrowse:GoUp())
},40,10,,,.T.)
    TButton():New(170,001,'GoDown()',oDlg,{|| oBrowse:GoDown())
},40,10,,,.T.)
    TButton():New(180,001,'GoTop()',oDlg,{|| oBrowse:GoTop())
},40,10,,,.T.)
    TButton():New(190,001,'GoBottom()', oDlg,{|| oBrowse:GoBottom())
},40,10,,,.T.)
    TButton():New(160,060,'nAt (Linha selecionada)',oDlg,;
                    {|| Alert(oBrowse:nAt)},80,10,,,.T.)
    TButton():New(170,060,'nRowCount (Nr de linhas visiveis)',oDlg,;
                    {|| Alert(oBrowse:nRowCount()) },80,10,,,.T.)
    TButton():New(180,060, 'nLen (Numero total de linhas)', oDlg,;
                    {|| Alert(oBrowse:nLen) },80,10,,,.T.)
    TButton():New( 190, 060, 'lEditCell (Edita a celula)', oDlg,;
                    {|| lEditCell(@aBrowse,oBrowse,'@!',3)
},40,10,,,.T.)

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL

.
```

## TSAY()

---

- Descrição:** Classe de objetos visuais do tipo controle – tSay, a qual exibe o conteúdo de texto estático sobre uma janela ou controle previamente definidos.

**Propriedades:**

IWordWrap	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.
ITransparent	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.

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

**Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
abText	Codeblock, opcional. Quando executado deve retornar uma cadeia de caracteres a ser exibida.
aoWnd	Objeto, opcional. Janela ou diálogo onde o controle será criado.
acPicture	Caractere, opcional. Picture de formatação do conteúdo a ser exibido.
aoFont	Objeto, opcional. Objeto tipo tFont para configuração do tipo de fonte que será utilizado para exibir o conteúdo.
IPar7	Reservado.
IPar8	Reservado.
IPar9	Reservado.
alPixels	Lógico, opcional. Se .T. considera coordenadas passadas em pixels se .F., padrão, considera as coordenadas passadas em caracteres.
anClrText	Numérico, opcional. Cor do conteúdo do controle.
anClrBack	Numérico, opcional. Cor do fundo do controle.
anWidth	Numérico, opcional. Largura do controle em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
IPar15	Reservado.
IPar16	Reservado.
IPar17	Reservado.
IPar18	Reservado.
IPar19	Reservado.

**Aparência:**



**Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg, oButton, oCombo, cCombo, cGet1:='Teste'
DEFINE MSDIALOG oDlg FROM 0,0 TO 300,300 PIXEL TITLE 'Meu test'
oSay:= tSay():New(01,01,{||'para exibir'},oDlg,,,
,,.T.,CLR_RED,CLR_WHITE,100,20)
ACTIVATE MSDIALOG oDlg CENTERED
Return
```



*Anotações*

---

---

---

---

## **TSCROLLBOX()**

---

- Descrição:** Classe de objetos visuais do tipo controle - tScrollbox, a qual permite criar um painel com scroll deslizantes nas laterais (horizontais e verticais) do controle.

- Propriedades:**

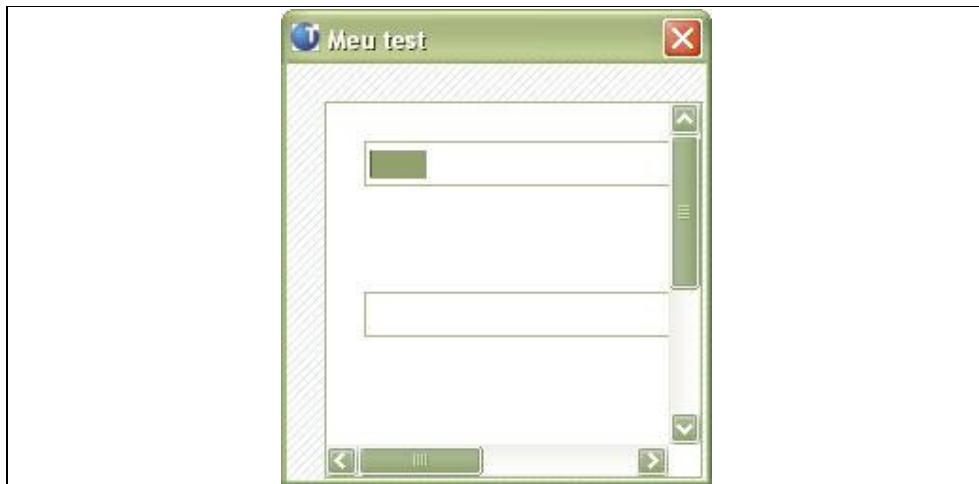
- Herdadas das classes superiores

- Construtor:** New([aoWnd], [anTop], [anLeft], [anHeight], [anWidth], [alVertical], [alHorizontal], [alBorder])

- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde o controle será criado.
anTop	Numérico, opcional. Coordenada vertical em pixels.
anLeft	Numérico, opcional. Coordenada horizontal em pixels.
anHeight	Numérico, opcional. Altura do controle em pixels.
anWidth	Numérico, opcional. Largura do controle em pixels.
alVertical	Lógico, opcional. Se .T. exibe a barra de scroll vertical.
alHorizontal	Lógico, opcional. Se .T. exibe a barra de scroll horizontal.
alBorder	Lógico, opcional. Se .T. exibe a borda do controle.

- Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg, oScr, oGet1, oGet2, oGet3
Local cGet1, cGet2, cGet3
cGet1:= Space(10)
cGet2:= Space(10)
cGet3:= Space(10)
DEFINE MSDIALOG oDlg FROM 0,0 TO 220,220 PIXEL TITLE 'Meu test'
  oScr:= TScrollBox():New(oDlg,10,10,100,100,.T.,.T.,.T.) // cria controles
  dentro do scrollbox
  @ 10,10 MSGET oGet1 VAR cGet1 SIZE 100,10 OF oScr PIXEL
  @ 50,10 MSGET oGet2 VAR cGet2 SIZE 100,10 OF oScr PIXEL
  @ 150,100 MSGET oGet3 VAR cGet3 SIZE 100,10 OF oScr PIXEL
ACTIVATE MSDIALOG oDlg CENTERED
Return
```

## **TSIMPLEEDITOR()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tSimpleEditor, a qual permite criar um controle visual para edição de textos com recursos simples, como o NotePad®

**Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([anRow], [anCol], [aoWnd], [anWidth], [anHeight],[ acText], [alReadOnly])

**Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
aoWnd	Janela ou controle onde o editor deverá ser criado.
anWidth	Numérico, opcional. Largura do editor em pixels.
anHeight	Numérico, opcional. Altura do editor em pixels.
acText	Texto, opcional. Inserido na inicialização do objeto.
alReadOnly	Lógico, opcional. Permite ou não a edição do texto

**Métodos auxiliares:**

### **Create**

---

- Descrição:** Método construtor opcional da classe.

- Sintaxe:** Create( aoWnd )

- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde o editor deverá ser criado.
-------	---

**Retorno:**

<b>Objeto</b>	Objeto do tipo TsimpleEditor.
---------------	-------------------------------

**Load**

---

**Descrição:** Carrega um texto para o editor.

**Sintaxe:** Load( acTexto )

**Parâmetros:**

acTexto	Texto. Texto que inicializará o editor.
---------	---

**Retorno:**

<b>Nenhum</b>	-
---------------	---

**TextBold**

---

**Descrição:** Texto em Negrito.

**Sintaxe:** TextBold( alBold )

**Parâmetros:**

alBold	Lógico. Habilita ou Desabilita a sessão do texto como Negrito.
--------	--

**Retorno:**

<b>Nenhum</b>	-
---------------	---

**TextUnderline**

---

**Descrição:** Texto Sublinhado.

**Sintaxe:** TextUnderline( alUnderline )

**Parâmetros:**

alUnderline	Lógico. Habilita ou Desabilita a sessão do texto Sublinhado.
-------------	--

**Retorno:**

<b>Nenhum</b>	-
---------------	---

**TextItalic**

---

**Descrição:** Texto Itálico.

**Sintaxe:** TextItalic( alItalic )

**Parâmetros:**

alItalic	Lógico. Habilita ou Desabilita a sessão do texto Itálico.
----------	---

**Retorno:**

<b>Nenhum</b>	-
---------------	---

## **TextFamily**

---

- Descrição:** Família de fontes.
- Sintaxe:** TextFamily( acFamily )
- Parâmetros:**

acFamily	Texto. Nome da família da fonte a ser usada na sessão do texto.
----------	---

- Retorno:**

Nenhum	-
--------	---

## **TextSize**

---

- Descrição:** Tamanho da fonte.
- Sintaxe:** TextSize( anSize )
- Parâmetros:**

anSize	Numérico. Tamanho da fonte utilizada na sessão do texto.
--------	--

- Retorno:**

Nenhum	-
--------	---

## **TextStyle**

---

- Descrição:** Estilo do parágrafo.
- Sintaxe:** TextStyle( anStyle )
- Parâmetros:**

anStyle	Numérico. Estilo do parágrafo a ser utilizada na sessão do texto.  1 – Normal 2 – Disco (Bullet) 3 – Circulo (Bullet) 4 – Quadrado (Bullet) 5 – Ordem decimal 6 – Ordem alfanumérica minúsculo 7 – Ordem alfanumérica maiúsculo
---------	---

- Retorno:**

Nenhum	-
--------	---

## **TextAlign**

---

- Descrição:** Alinhamento do texto.
- Sintaxe:** TextAlign( anAlign )
- Parâmetros:**

anAlign	Numérico. Tipo do alinhamento do parágrafo. 1 – Esquerda 2 – Direita 3 – Centralizado 4 – Justificado
---------	---

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **TextFormat**

---

- Descrição:** Formato do texto.
- Sintaxe:** TextFormat( anFormat )
- Parâmetros:**

anFormat	Numérico. Formato do texto 1 – Html 2 – Plain Text
----------	--

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **RetText**

---

- Descrição:** Retorna o texto em formato string.
- Sintaxe:** RetText( void )
- Parâmetros:**

void	Retorna uma string com o conteúdo do editor
------	---

- Retorno:**

<b>Nenhum</b>	-
---------------	---

## **RetTextSel**

---

- Descrição:** Retorna o texto selecionado em formato string..
- Sintaxe:** RetText()
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

<b>String</b>	Texto selecionado.
---------------	--------------------

## **TextStatus**

---

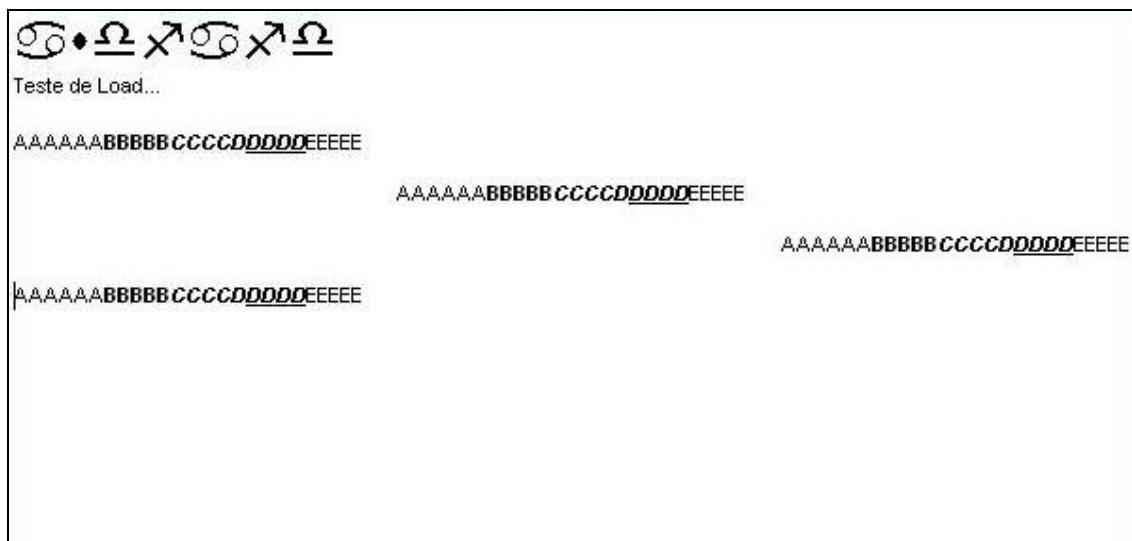
- Descrição:** Retorna um array com as propriedades do texto posicionado.
- Sintaxe:** TextStatus()
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

<b>Array</b>	Retorna um array com as seguintes propriedades : 1 – Lógico Negrito 2 – Lógico Itálico 3 – Lógico Sublinhado 4 – Caracter RGB da cor 5 – Caracter Tamanho da fonte 6 – Caracter Fonte
--------------	---

- Aparência:**



## **Exemplo:**

---

```
#include "protheus.ch"

User Function MyEditor()
Local oDlg, oEditor
DEFINE MSDIALOG oDlg FROM 0,0 TO 500,600 PIXEL TITLE "Meu Editor"
// Usando o método create
oEdit := tSimpleEditor():Create( oDlg )
oEdit:nTop      := 10
oEdit:nLeft     := 10
oEdit:nWidth    := 600
oEdit:nHeight   := 500

// Usando o método new
oEdit := tSimpEdit():New( 0, 0, oDlg, 500, 600 )

ACTIVATE MSDIALOG oDlg CENTERED
Return Nil
```

## **TSLIDER()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tSlider, a qual permite criar um controle visual do tipo botão deslizante.

**Propriedades:**

bChange	Bloco de código. Executado toda vez que o valor é alterado retornando o novo valor.
---------	---

- Construtor:** New([anRow], [anCol], [aoWnd], [abChange], [anWidth], [anHeight], [acMsg], [abWhen])

**Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels
anCol	Numérico, opcional. Coordenada horizontal em pixels
aoWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
abChange	Bloco de código, opcional. Bloco que deverá ser acionado quando o botão for movimentado.
anWidth	Numérico, opcional. Largura do botão em pixels.
anHeight	Numérico, opcional. Altura do botão em pixels.
acMsg	Caracter, opcional. Mensagem de hint do botão
abWhen	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

**Métodos auxiliares:**

## Create

---

- Descrição:** Método construtor da classe.
- Sintaxe:** Create( aoWnd )
- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
-------	--

- Retorno:**

Objeto	Objeto do tipo TSlide criado.
--------	-------------------------------

## setRange

---

- Descrição:** Especifica a faixa de valores.
- Sintaxe:** setRange( min, max )
- Parâmetros:**

Min	Numérico. Valor mínimo do botão.
Max	Numérico. Valor máximo do botão.

- Retorno:**

Nenhum	-
--------	---

## setMarks

---

- Descrição:** Especifica o tipo de marcação do botão.
- Sintaxe:** setMarks( nTipo )
- Parâmetros:**

nTipo	Numérico. Tipo de marcação do botão. 1 -
Max	Numérico. Valor máximo do botão. 0 – Sem marcação 1 – Acima (esquerda se vertical) 2 – Abaixo (direita se vertical) 3 – Ambos os lados

- Retorno:**

Nenhum	-
--------	---

## **setInterval**

---

- Descrição:** Especifica a distância entre um marcador e outro.
- Sintaxe:** setInterval( nInterval )
- Parâmetros:**

nInterval	Numérico. Valor entre os marcadores
-----------	-------------------------------------

- Retorno:**

Nenhum	-
--------	---

## **setValue**

---

- Descrição:** Especifica um valor para o botão.
- Sintaxe:** setValue( nVal )
- Parâmetros:**

nVal	Numérico. Valor do botão
------	--------------------------

- Retorno:**

Nenhum	-
--------	---

## **setStep**

---

- Descrição:** Especifica o valor dos passos do botão.
- Sintaxe:** setStep( nStep )
- Parâmetros:**

nStep	Numérico. Valor do passo do botão
-------	-----------------------------------

- Retorno:**

Nenhum	-
--------	---

## **setOrient**

---

- Descrição:** Especifica a orientação do botão, horizontal ou vertical.
- Sintaxe:** setOrient( nOrient )
- Parâmetros:**

nOrient	Numérico. Orientação do botão 0 – Horizontal 1 – Vertical
---------	---

- Retorno:**

Nenhum	-
--------	---

**Aparência:**



**Exemplo:**

```
#include "protheus.ch"
#include "hbutton.ch"
User Function MytSlider()
Local oDlg, oSlider
DEFINE MSDIALOG oDlg FROM 0,0 TO 500,600 PIXEL TITLE "Meu tSlider"
// Usando o método create
oSlider:= tSlider():Create( oDlg )
oSlider:nTop      := 100
oSlider:nLeft     := 10
oSlider:nWidth    := 100
oSlider:nHeight   := 30

// Usando o command
@ 100, 10 SLIDER oSlider SIZE 30, 100 OF oDlg MESSAGE 'tSlider'
ACTIVATE MSDIALOG oDlg CENTERED
Return Nil
```

**TSPLITTER()**

- Descrição:** Classe de objetos visuais do tipo controle – tSplitter, a qual permite criar um controle visual do tipo divisor.

**Propriedades:**

-	Herdadas as classes superiores.
---	---------------------------------

- Construtor:** New( [anRow], [anCol], [aoWnd], [anWidth], [anHeight], [anOrientation] )

**Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical
anCol	Numérico, opcional. Coordenada horizontal
aoWnd	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
anWidth	Numérico, opcional. Largura do objeto
anHeight	Numérico, opcional. Altura do objeto
anOrientation	Numérico, opcional. Sentido no qual deverão ser criado os divisores. 0 – Horizontal 1 – Vertical

**Métodos auxiliares:**

**Create**

---

- Descrição:** Método construtor da classe.
- Sintaxe:** Create( aoWnd )
- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
-------	--

- Retorno:**

Objeto	Objeto do tipo Tsplitter gerado.
--------	----------------------------------

**setOrient**

---

- Descrição:** Especifica a orientação do divisor, horizontal ou vertical.
- Sintaxe:** setOrient( nOrient )
- Parâmetros:**

nOrient	Numérico. Orientação do botão 0 – Horizontal 1 – Vertical
---------	---

- Retorno:**

Nenhum	-
--------	---

**setChildCollapse**

---

- Descrição:** Especifica se os elementos podem ser “collapsibles”.
- Sintaxe:** setChildCollapse( IColl )
- Parâmetros:**

IColl	Lógico. Ativa / Desativa
-------	--------------------------

- Retorno:**

Nenhum	-
--------	---

## **setCollapse**

---

- Descrição:** Especifica o objeto que pode ser "collapsible".
- Sintaxe:** setCollapse( oObj, IColl )
- Parâmetros:**

oObj	Objeto. Controle a ser "collapsed"
IColl	Lógico. Ativa / Desativa

- Retorno:**

Nenhum	-
--------	---

## **movToLast**

---

- Descrição:** Coloca o objeto como ultimo das divisões.
- Sintaxe:** movToLast( oObj )
- Parâmetros:**

oObj	Objeto. Controle a posicionado
------	--------------------------------

- Retorno:**

Nenhum	-
--------	---

## **movToFirst**

---

- Descrição:** Coloca o objeto como primeiro das divisões.
- Sintaxe:** movToFirst( oObj )
- Parâmetros:**

oObj	Objeto. Controle a ser posicionado
------	------------------------------------

- Retorno:**

Nenhum	-
--------	---

## **setOpaqueResize**

---

- Descrição:** Especifica se o resize deve ser opaco.
- Sintaxe:** setOpaqueResize( IOpaq )
- Parâmetros:**

IOpaq	Lógico. Ativa / Desativa
-------	--------------------------

- Retorno:**

Nenhum	-
--------	---

## **setResizeMode**

---

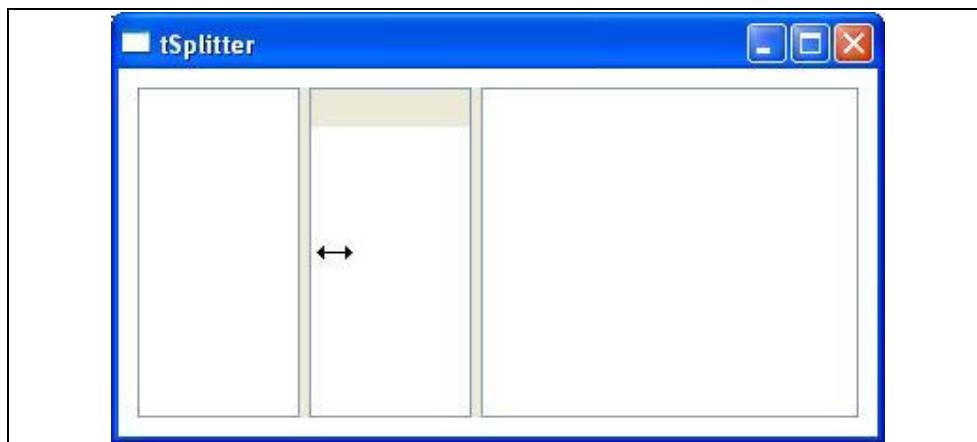
- Descrição:** Especifica o modo de "resize" do objeto..
- Sintaxe:** **setResizeMode** ( oObj, nMode )
- Parâmetros:**

<b>oObj</b>	Objeto. Controle a ser "resized"
<b>nMode</b>	Numérico. Modo do "resize" 0 – Stretch 1- KeepSize 2 – FollowSizeHint 3 – Auto

- Retorno:**

<b>Nenhum</b>	-
---------------	---

- Aparência:**



*Anotações*

---

---

---

---

---

## **Exemplo:**

---

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

User Function MytSplitter()
Local oDlg, oSplitter, oPanel1, oPanel2, oPanel3

DEFINE MSDIALOG oDlg FROM 0,0 TO 500,600 PIXEL TITLE "Meu tSplitter"

// Usando o método create
oSplitter := tSplitter():Create( oDlg )
@ 1, 1 MSPANEL oPanel1 OF oSplitter
@ 1, 2 MSPANEL oPanel2 OF oSplitter
@ 1, 3 MSPANEL oPanel3 OF oSplitter

// Usando o command
@ 1, 1 SPLITTER oSplitter SIZE 100, 100 OF oDlg
@ 1, 1 MSPANEL oPanel1 OF oSplitter
@ 1, 2 MSPANEL oPanel2 OF oSplitter
@ 1, 3 MSPANEL oPanel3 OF oSplitter

ACTIVATE MSDIALOG oDlg CENTERED

Return Nil
```

---

## **TTABS()**

---

- Descrição:** Classe de objetos visuais do tipo controle – TTabs, a qual permite criar um controle visual do tipo pasta.

**Propriedades:**

-	Herdadas das classes superiores.
---	----------------------------------

- Construtor:** New([anTop], [anLeft], [aPrompts], [bAction], [oWnd], [nOption], [nClrFore], [nClrBack], [IPixel], [IDesign], [nWidth], [nHeighth], [cMsg])

**Parâmetros:**

anTop	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anLeft	Numérico, opcional. Coordenada horizontal em pixels ou caracteres
aPrompts	Array, Titulo dos folders
bAction	Bloco de código, Disparado na troca da aba.
oWnd	Objeto, opcional. Janela ou controle onde a aba deverá ser criado
nOption	Numérico, opcional. Folder selecionado
nClrFore	Numérico, opcional. Cor de frente
nClrBack	Numérico, opcional. Cor de fundo

IPixel	Lógico, opcional. Utiliza coordenadas em pixel
IDesign	Lógico, opcional. NÃO USADO
nWidth	Numérico, opcional. Largura em pixels
nHeigth	Numérico, opcional. Altura em pixels.
cMsg	Caractere, Mensagem de Hint

**Métodos auxiliares:**

**AddItem**

---

- Descrição:** Adiciona uma aba na pasta
- Sintaxe:** AddItem(cTítulo)
- Parâmetros:**

cTítulo	Título da aba que será adicionada
---------	-----------------------------------

- Retorno:**

Nenhum	-
--------	---

**SetOption**

---

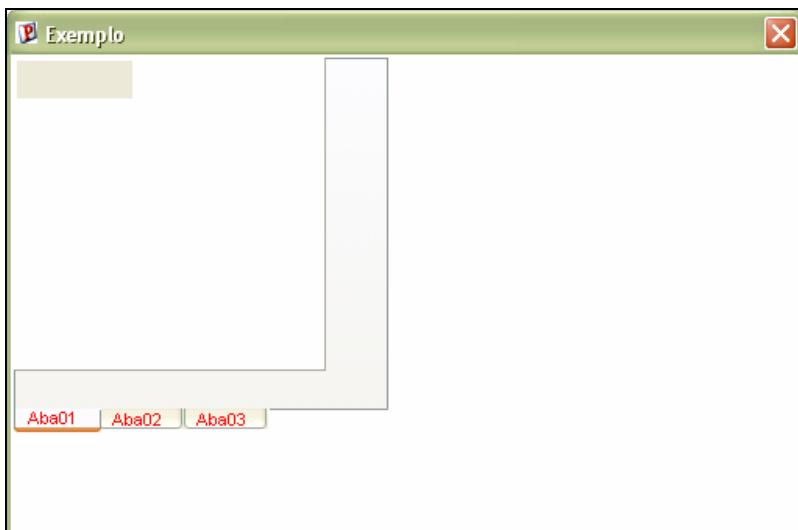
- Descrição:** Seleciona a Aba
- Sintaxe:** SetOption( nNrAba )
- Parâmetros:**

nNrAba	Número que identifica a posição da aba que será selecionada.
--------	--

- Retorno:**

Nenhum	-
--------	---

**Aparência:**



### **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
    DEFINE MSDIALOG oDlg FROM 0,0 TO 304,504 PIXEL TITLE 'Exemplo'

    oTTabs := TTabs():New(01,01,{'Aba01','Aba02','Aba03'},;
        {||oPanel01:LVISIBLECONTROL:=(oTTabs:nOption==1)},;
        oDlg,,RGB(255,0,0),RGB(255,255,0),.T.,,120,120,)
    oPanel01 := TPanel():New( 000, 000,'',oTTabs,,,,,100,100,,.T. )
    oPanel01:lVisibleControl := .T.
    oBtn01 := TButton():New( 01,01,'TButton01',oPanel01,;
        {||oTTabs:SetOption(2)}, 037,
012,,,F.,,F.,,F.,,,F. )

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

---

### **TTOOLBOX()**

---

- Descrição:** Classe de objetos visuais do tipo controle – tToolbox, a qual permite criar um controle visual para agrupar diferentes objetos.

**Propriedades:**

bChangeGrp	Bloco de código. Executado na troca entre os grupos existentes.
------------	---

- Construtor:** New([anRow], [anCol], [aoWnd], [anWidth], [anHeight], [aoFont], [acMsg]], [abWhen])

**Parâmetros:**

anRow	Numérico, opcional. Coordenada vertical em pixels ou caracteres.
anCol	Numérico, opcional. Coordenada horizontal em pixels ou caracteres.
aoWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
anWidth	Numérico, opcional. Largura do botão em pixels.
anHeight	Numérico, opcional. Altura do botão em pixels.
aoFont	Objeto, opcional. Objeto tipo tFont com propriedades da fonte utilizada para o título do botão.
acMsg	Mensagem, opcional. Tooltip/Hint do componente.
abWhen	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

**Métodos auxiliares:**

**Create**

---

- Descrição:** Método construtor opcional da classe.
- Sintaxe:** Create( aoWnd )
- Parâmetros:**

aoWnd	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
-------	--

- Retorno:**

Objeto	Objeto do tipo TtoolBox gerado.
--------	---------------------------------

**addGroup**

---

- Descrição:** Adiciona um grupo
- Sintaxe:** addGroup( aoObj, acName, aoIcon )
- Parâmetros:**

aoObj	Objeto. Objeto Pai que vai ser inserido no grupo.
acName	Caractere. Descrição do grupo
aoIcon	Objeto, opcional. Ícone para o grupo

- Retorno:**

Nenhum	-
--------	---

**removeGroup**

---

- Descrição:** Remove um grupo
- Sintaxe:** removeGroup( aoObj )
- Parâmetros:**

aoObj	Objeto. Objeto Pai que vai ser removido do grupo.
-------	---

- Retorno:**

Nenhum	-
--------	---

**setCurrentGroup**

---

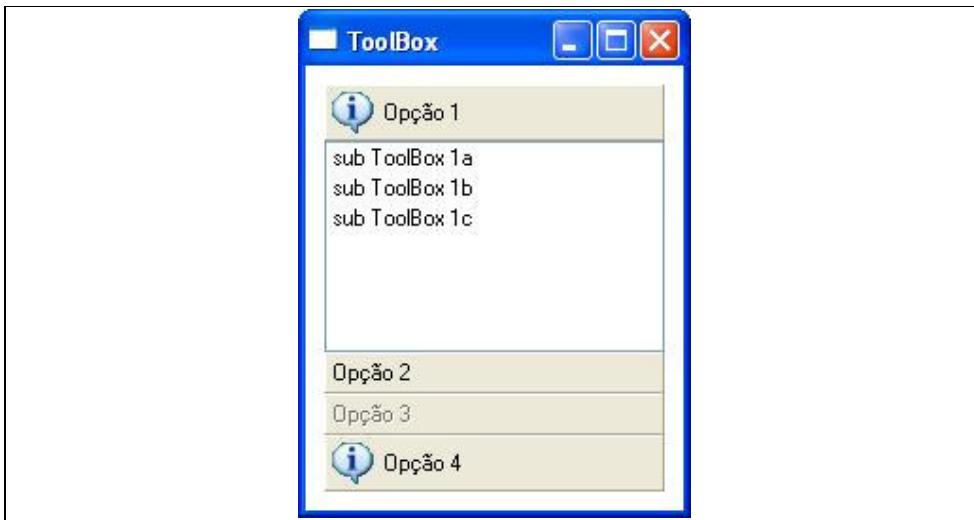
- Descrição:** Define o grupo corrente
- Sintaxe:** setCurrentGroup( aoObj )
- Parâmetros:**

aoObj	Objeto. Objeto Pai que será definido como grupo corrente.
-------	---

- Retorno:**

Nenhum	-
--------	---

**Aparência:**



**Exemplo:**

```
#include "protheus.ch"

User Function MytToolbox()
Local oDlg, oTb

DEFINE MSDIALOG oDlg FROM 0,0 TO 500,600 PIXEL TITLE "Meu Toolbox"

// Usando o método create
oTb:= tToolbox():Create( oDlg )
oTb:nTop      := 100
oTb:nLeft     := 10
oTb:nWidth    := 100
oTb:nHeight   := 30

oTb:Add( oPanel, "Opção 1", oIcone)

// Usando o command
@ 200,100 TOOLBOX oTb SIZE 100,30 OF oDlg

TOOLBOX oTb ADDGROUP TITLE "Opção 1" OBJECT oPanel ICON oIcone

ACTIVATE MSDIALOG oDlg CENTERED

Return Nil
```

## TWBROWSE()

---

- Descrição:** Classe de objetos visuais do tipo controle - TWBrowse, a qual permite criar um controle visual do tipo Grid.

**Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>nAt</b>	Linha atualmente selecionada / posicionada no objeto
<b>nLen</b>	Número total de linhas do objeto

- Construtor:** New([nRow], [nCol], [nWidth], [nHeighth],[bFields], [aHeaders], [aColSizes], [oDlg], [cField], [uValue1], [uValue2], [uChange],[{|nRow,nCol,nFlags|[uLDbClick]}], [{|nRow,nCol,nFlags|[uRClick]}], [oFont], [oCursor], [nClrFore], [nClrBack], [cMsg], [IUpdate], [cAlias], [IPixel], [{uWhen}], [IDesign], [bValid], [IHSscroll], [IVScroll])

**Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical
<b>nCol</b>	Numérico, opcional. Coordenada horizontal
<b>nWidth</b>	Numérico, opcional. Largura do objeto
<b>nHeight</b>	Numérico, opcional. Altura do objeto
<b>bFields</b>	Bloco de código, Lista de Campos
<b>aHeaders</b>	Vetor, Descrição dos campos para no cabeçalho
<b>aColSizes</b>	Vetor, Largura das colunas
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
<b>cField</b>	Caracter, opcional. Campos necessários ao filtro
<b>uValue1</b>	Indefinido, opcional. Inicio do intervalo para o filtro
<b>uValue2</b>	Indefinido, opcional. Fim do intervalo para o filtro
<b>bChange</b>	Bloco de código, opcional. Executado quando o item selecionado é alterado.
<b>bLDbClick</b>	Bloco de código, opcional. Executado quando acionado duplo click do botão esquerdo do mouse sobre o controle.
<b>bRClick</b>	Não utilizado
<b>oFont</b>	Objeto, opcional. Fonte
<b>oCursor</b>	Objeto, opcional. Tipo do Cursor
<b>nClrFore</b>	Numérico, opcional. Cor do texto da janela.
<b>nClrBack</b>	Numérico, opcional. Cor de fundo da janela.
<b>cMsg</b>	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
<b>IUpdate</b>	Não utilizado
<b>cAlias</b>	Caracter, opcional se objeto utilizado com Vetor, obrigatório se utilizado com Tabela
<b>IPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.
<b>IDesign</b>	Não Utilizado

bValid	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.
IHScroll	Lógico, opcional. Se .T., habilita barra de rolagem horizontal.
IVScroll	Lógico, opcional. Se .T., habilita barra de rolagem vertical.

**Métodos auxiliares:**

#### GoUp

- Descrição:** Salta uma linha para cima.
- Sintaxe:** GoUp( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

#### GoDown

- Descrição:** Salta uma linha para baixo.
- Sintaxe:** GoDown( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

#### GoTop

- Descrição:** Salta para primeira linha.
- Sintaxe:** GoTop( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

#### GoBottom

- Descrição:** Salta para ultima linha.
- Sintaxe:** GoBottom( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil	
-----	--

### RowCount

- Descrição:** Retorna numero de linhas visiveis.
- Sintaxe:** RowCount( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil	
-----	--

### LEditCell

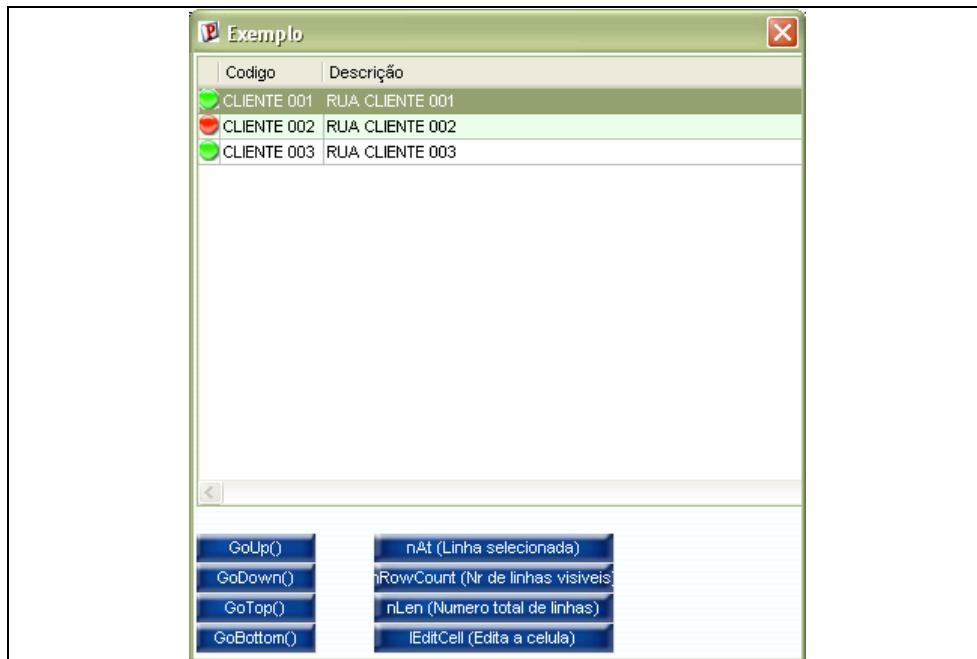
- Descrição:** Edita o valor de uma coluna.
- Sintaxe:** IEditCell( aList, oList, cPicture, nCol )
- Parâmetros:**

aList	Vetor da Browse onde estão os valores da mesma
oList	Objeto, Browse a ser editado
cPicture	Caracter, picture necessária para edição do campo
nCol	Numérico, coluna a ser editada.

- Retorno:**

Nil	
-----	--

**Aparência:**



## **Exemplo:**

---

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
Local oOK := LoadBitmap(GetResources(),'br_verde')
Local oNO := LoadBitmap(GetResources(),'br_vermelho')

    DEFINE MSDIALOG oDlg FROM 0,0 TO 402,402 PIXEL TITLE 'Exemplo'

    oBrowse := TWBrowse():New( 01 , 01, 245, 150,,;
        {'','Codigo','Descrição'}, {20,30,30}, oDlg, ,,,,
        {|||},.....F.,..T.,..F.,.. )
    aBrowse := {{.T.,'CLIENTE 001','RUA CLIENTE 001','BAIRRO CLIENTE
001'},;
                {.F.,'CLIENTE 002','RUA CLIENTE 002','BAIRRO CLIENTE
002'},;
                {.T.,'CLIENTE 003','RUA CLIENTE 003','BAIRRO CLIENTE
003'} }

    oBrowse:SetArray(aBrowse)
    oBrowse:bLine := {||{;
        If(aBrowse[oBrowse:nAt,01],oOK,oNO),;
        aBrowse[oBrowse:nAt,02],;
        aBrowse[oBrowse:nAt,03],;
        aBrowse[oBrowse:nAt,04] } }
    oBrowse:bLDbClick := ;
    {|| aBrowse[oBrowse:nAt][1] :=
    !aBrowse[oBrowse:nAt][1],oBrowse:DrawSelect()}

    // Principais comandos
    TButton():New(160,001,'GoUp()',oDlg,{|| oBrowse:GoUp()
},40,10,,,.T.)
    TButton():New(170,001,'GoDown()',oDlg,{|| oBrowse:GoDown()
},40,10,,,.T.)
    TButton():New(180,001,'GoTop()',oDlg,{|| oBrowse:GoTop()
},40,10,,,.T.)
    TButton():New(190,001,'GoBottom()', oDlg,{|| oBrowse:GoBottom()
},40,10,,,.T.)
    TButton():New(160,060,'nAt (Linha selecionada)',oDlg,;
        {|| Alert(oBrowse:nAt)},80,10,,,.T.)
    TButton():New(170,060,'nRowCount (Nr de linhas visiveis)',oDlg,;
        {|| Alert(oBrowse:nRowCount()) },80,10,,,.T.)
    TButton():New(180,060,'nLen (Numero total de linhas)', oDlg,;
        {|| Alert(oBrowse:nLen) },80,10,,,.T.)
    TButton():New(190,060,'lEditCell (Edita a celula)', oDlg,;
        {|| lEditCell(@aBrowse,oBrowse,'@!',3)
},80,10,,,.T.)

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## VCBROWSE()

---

- Descrição:** Classe de objetos visuais do tipo controle – VCBrowse, a qual permite criar um controle visual do tipo Grid.

**Propriedades:**

<b>+</b>	Herdadas da classe superior
<b>nAt</b>	Linha atualmente selecionada / posicionada no objeto
<b>nLen</b>	Número total de linhas do objeto

- Construtor:** New([nRow], [nCol], [nWidth], [nHeighth],[bFields], [aHeaders], [aColSizes], [oDlg], [cField], [uValue1], [uValue2], [uChange],[{|nRow,nCol,nFlags|}[uLDbClick]]}, [|{|nRow,nCol,nFlags|}[uRClick]}], [oFont], [oCursor], [nClrFore], [nClrBack], [cMsg], [IUpdate], [cAlias], [IPixel], [{uWhen}], [IDesign], [bValid], [IHScroll], [IVScroll])

**Parâmetros:**

<b>nRow</b>	Numérico, opcional. Coordenada vertical
<b>nCol</b>	Numérico, opcional. Coordenada horizontal
<b>nWidth</b>	Numérico, opcional. Largura do objeto
<b>nHeight</b>	Numérico, opcional. Altura do objeto
<b>bFields</b>	Bloco de código, Lista de Campos
<b>aHeaders</b>	Vetor, Descrição dos campos para no cabeçalho
<b>aColSizes</b>	Vetor, Largura das colunas
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o divisor deverá ser criado.
<b>cField</b>	Caracter, opcional. Campos necessários ao filtro
<b>uValue1</b>	Indefinido, opcional. Inicio do intervalo para o filtro
<b>uValue2</b>	Indefinido, opcional. Fim do intervalo para o filtro
<b>bChange</b>	Bloco de código, opcional. Executado quando o item selecionado é alterado.
<b>bLDbClick</b>	Bloco de código, opcional. Executado quando acionado duplo click do botão esquerdo do mouse sobre o controle.
<b>bRClick</b>	Não utilizado
<b>oFont</b>	Objeto, opcional. Fonte
<b>oCursor</b>	Objeto, opcional. Tipo do Cursor
<b>nClrFore</b>	Numérico, opcional. Cor do texto da janela.
<b>nClrBack</b>	Numérico, opcional. Cor de fundo da janela.
<b>cMsg</b>	Caracter, opcional. Mensagem ao posicionar o mouse sobre o objeto
<b>IUpdate</b>	Não utilizado
<b>cAlias</b>	Caracter, opcional se objeto utilizado com Vetor, obrigatório se utilizado com Tabela
<b>IPixel</b>	Lógico, opcional. Se .T. as coordenadas informadas são em pixels, se .F. são em caracteres.
<b>bWhen</b>	Bloco de código, opcional. Executado quando mudança de foco de entrada de dados está sendo efetuada na janela onde o controle foi criado. O bloco deve retornar .T. se o controle deve permanecer habilitado ou .F. se não.

IDesign	Não Utilizado
bValid	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.
IHScroll	Lógico, opcional. Se .T., habilita barra de rolagem horizontal.
IVScroll	Lógico, opcional. Se .T., habilita barra de rolagem vertical.

**Métodos auxiliares:**

### **GoUp**

---

- Descrição:** Salta uma linha para cima.
- Sintaxe:** GoUp( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

### **GoDown**

---

- Descrição:** Salta uma linha para baixo.
- Sintaxe:** GoDown( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

### **GoTop**

---

- Descrição:** Salta para primeira linha.
- Sintaxe:** GoTop( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil
-----

## **GoBottom**

---

- Descrição:** Salta para ultima linha.
- Sintaxe:** GoBottom( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil	
-----	--

## **RowCount**

---

- Descrição:** Retorna numero de linhas visiveis.
- Sintaxe:** RowCount( )
- Parâmetros:**

Nenhum	-
--------	---

- Retorno:**

Nil	
-----	--

## **LEditCell**

---

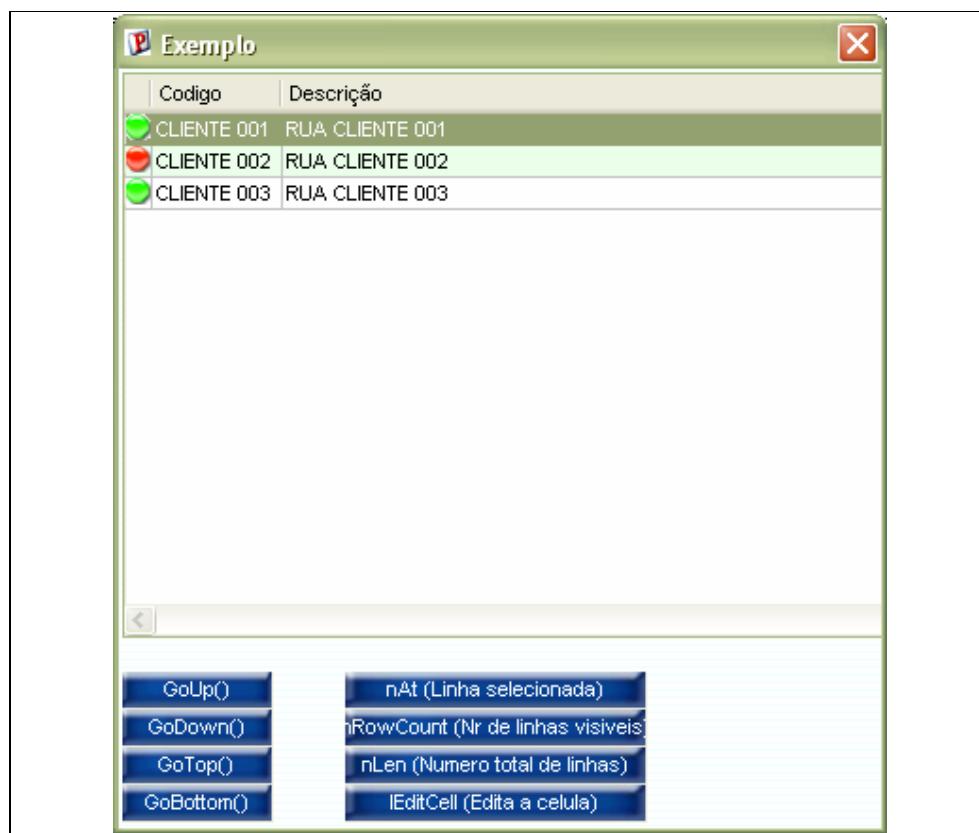
- Descrição:** Edita o valor de uma coluna.
- Sintaxe:** LEeditCell( aList, oList, cPicture, nCol )
- Parâmetros:**

aList	Vetor da Browse onde estão os valores da mesma
oList	Objeto, Browse a ser editado
cPicture	Caracter, picture necessária para edição do campo
nCol	Numérico, coluna a ser editada.

- Retorno:**

Nil	
-----	--

**Aparência:**



*Anotações*

---

---

---

---

## **Exemplo:**

```
#include 'protheus.ch'
User Function Teste()
Local oDlg
Local oOK := LoadBitmap(GetResources(),'br_verde')
Local oNO := LoadBitmap(GetResources(),'br_vermelho')
    DEFINE MSDIALOG oDlg FROM 0,0 TO 402,402 PIXEL TITLE 'Exemplo'

    oBrowse := VCBrowse():New( 01 , 01, 245, 150,,;
        {'','Codigo','Descrição'}, {20,30,30}, oDlg, ....;
        {|||},.....F...,T...,F.... )
    aBrowse := {{.T.,'CLIENTE 001','RUA CLIENTE 001','BAIRRO CLIENTE
001'};;
                {.F.,'CLIENTE 002','RUA CLIENTE 002','BAIRRO CLIENTE
002'};;
                {.T.,'CLIENTE 003','RUA CLIENTE 003','BAIRRO CLIENTE
003'} }

    oBrowse:SetArray(aBrowse)
    oBrowse:bLine := {||{;
        If(aBrowse[oBrowse:nAt,01],oOK,oNO),;
        aBrowse[oBrowse:nAt,02],;
        aBrowse[oBrowse:nAt,03],;
        aBrowse[oBrowse:nAt,04] } }
    oBrowse:bLDblClick :=;
        {|| aBrowse[oBrowse:nAt][1] :=
    !aBrowse[oBrowse:nAt][1],oBrowse:DrawSelect()}

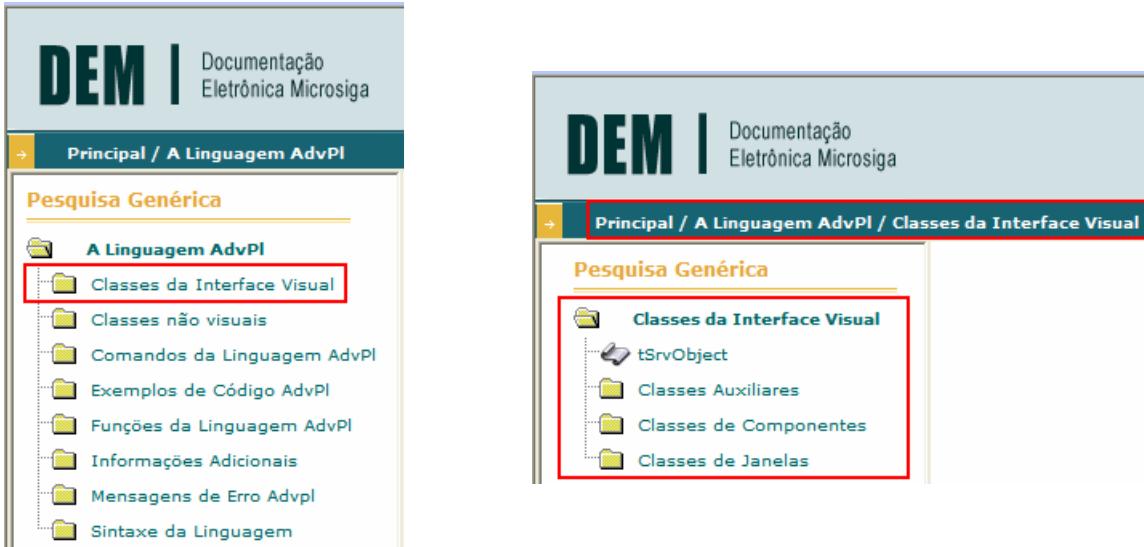
    // Principais comandos
    TButton():New(160,001,'GoUp()',oDlg,{|| oBrowse:GoUp()
},40,10,,,.T.)
        TButton():New(170,001,'GoDown()',oDlg,{|| oBrowse:GoDown()
},40,10,,,.T.)
        TButton():New(180,001,'GoTop()',oDlg,{|| oBrowse:GoTop()
},40,10,,,.T.)
        TButton():New(190,001,'GoBottom()' , oDlg,{|| oBrowse:GoBottom()
},40,10,,,.T.)
        TButton():New(160,060,'nAt (Linha selecionada)',oDlg,;
            {|| Alert(oBrowse:nAt)},80,10,,,.T.)
        TButton():New(170,060,'nRowCount (Nr de linhas visiveis)',oDlg,;
            {|| Alert(oBrowse:nRowCount()) },80,10,,,.T.)
        TButton():New(180,060, 'nLen (Numero total de linhas)', oDlg,;
            {|| Alert(oBrowse:nLen) },80,10,,,.T.)
        TButton():New(190,060, 'lEditCell (Edita a celula)', oDlg,;
            {|| lEditCell(@aBrowse,oBrowse,'@!',3)
},40,10,,,.T.)

    ACTIVATE MSDIALOG oDlg CENTERED
Return NIL
```

## Documentação dos componentes da interface visual

Os componentes da interface visual da linguagem ADVPL utilizados neste treinamento estão documentados na seção Guia de Referência, ao final deste material.

Para visualizar a documentação completa de todos os componentes mencionados neste capítulo deve ser acesso o site DEM – Documentação Eletrônica Microsiga ([dem.microsiga.com.br](http://dem.microsiga.com.br)) conforme abaixo:



Anotações

---

---

---

---

## 4.1. Particularidades dos componentes visuais

### 4.1.1. Configurando as cores para os componentes

Os componentes visuais da linguagem ADVPL utilizam o padrão de cores RGB.

As cores deste padrão são definidas pela seguinte fórmula, a qual deve ser avaliada tendo como base a paleta de cores no formato RGB:

```
nCor := nVermelho + (nVerde * 256) + (nAzul * 65536)
```

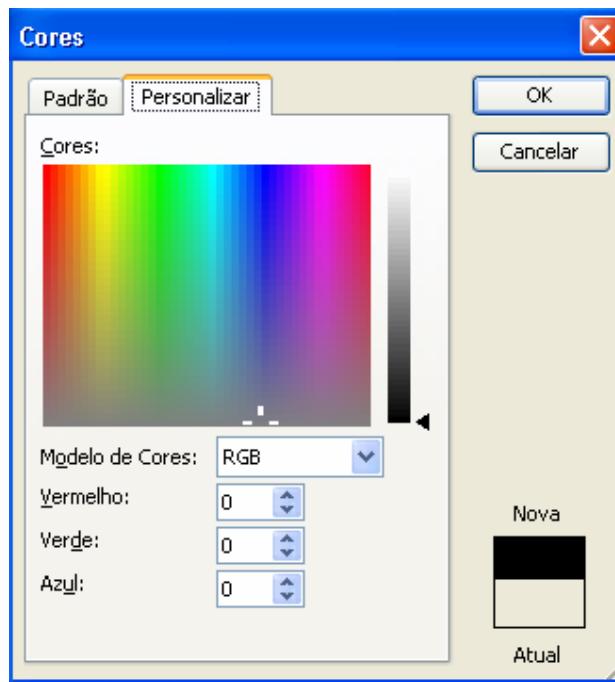


Figura: Paleta de cores no formato RGB

Com base nesta paleta, podemos definir os valores das seguintes cores básicas:

Cor	R	G	B	Valor
Preto	0	0	0	0
Azul	0	0	255	16711680
Verde	0	255	0	65280
Ciano	0	255	255	16776960
Vermelho	255	0	0	255
Rosa	255	0	255	16711935
Amarelo	255	255	0	65535
Branco	255	255	255	16777215

Para atribuir as cores aos objetos visuais devem ser observados os atributos utilizados para estes fins em cada objeto, como por exemplo:

#### **MSDIALOG()**

---

<b>nClrPane</b>	Cor de fundo do painel
<b>nClrText</b>	Cor da fonte das letras do painel

#### **TSAY()**

---

<b>nClrPane</b>	Cor de fundo do painel
<b>nClrText</b>	Cor da fonte das letras do painel

#### **Função RGB()**

---

A linguagem ADVPL possui a função RGB() a qual retorna o valor da cor a ser definido, de acordo com a parametrização de cada um dos elementos da paleta RGB.

#### **RGB(nRed, nGreen, nBlue)**

---

<b>nRed</b>	Valor de 0-255 para o elemento vermelho da paleta RGB
<b>nGreen</b>	Valor de 0-255 para o elemento verde da paleta RGB
<b>nBlue</b>	Valor de 0-255 para o elemento azul da paleta RGB
<b>Retorno</b>	Valor a ser definido para o atributo cor do componente

## 5. Aplicações com a interface visual do ADVPL

A linguagem ADVPL possui interfaces visuais pré-definidas que auxiliam no desenvolvimento de aplicações mais completas, combinando estas interfaces com os componentes visuais demonstrados anteriormente.

Didaticamente as interfaces visuais pré-definidas da linguagem ADVPL podem ser divididas em três grupos:

- Captura de informações simples ou Multi-Gets;**
- Captura de múltiplas informações ou Multi-Lines;**
- Barras de botões**

### 5.1. Captura de informações simples (Multi-Gets)

Em ADVPL, as telas de captura de informações compostas por múltiplos campos digitáveis acompanhados de seus respectivos textos explicativos são comumente chamados de Enchoices.

Um Enchoice pode ser facilmente entendida como diversos conjuntos de objetos TSay e TGet alinhados de forma a visualizar ou capturar informações, normalmente vinculadas a arquivos de cadastros ou movimentações simples.

Abaixo temos a visualização de uma Enchoice para o arquivo padrão do ERP Protheus de Cadastro de Clientes ("SA1"):

**Figura: Enchoice do Cadastro de Clientes do ERP Protheus**

A linguagem ADVPL permite a implementação da Enchoice de duas formas similares:

- Função Enchoice:** Sintaxe tradicionalmente utilizada em ADVPL, a qual não retorna um objeto para a aplicação chamadora;
- Classe MsMGet:** Classe do objeto Enchoice, a qual permite a instanciação direta de um objeto, tornando-o disponível na aplicação chamadora.

A utilização de um ou outro objeto depende unicamente da escolha do desenvolvedor já que os parâmetros para a função Enchoice e para o método New() da classe MsMGet são os mesmos, lembrando que para manter a coerência com uma aplicação escrita em orientação a objetos deverá ser utilizada a classe MsMGet().

### 5.1.1. Enchoice()

- Sintaxe:** Enchoice( **cAlias**, **nReg**, **nOpc**, **aCRA**, **cLetra**, **cTexto**, **aAcho**, **aPos**, **aCpos**, **nModelo**, **nColMens**, **cMensagem**, **cTudoOk**, **oWnd**, **IF3**, **IMemoria**, **IColumn**, **caTela**, **INoFolder**, **IProperty**)
- Retorno:** Nil
- Parâmetros:**

<b>cAlias</b>	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
<b>nReg</b>	Parâmetro não utilizado
<b>nOpc</b>	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização)
<b>aCRA</b>	Parâmetro não utilizado
<b>cLetra</b>	Parâmetro não utilizado
<b>cTexto</b>	Parâmetro não utilizado
<b>aAcho</b>	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"
<b>aPos</b>	Vetor com coordenadas para criação da enchoice no formato {<top>, <left>, <bottom>, <right>}
<b>aCpos</b>	Vetor com nome dos campos que poderão ser editados
<b>nModelo</b>	Se for diferente de 1 desabilita execução de gatilhos estrangeiros
<b>nColMens</b>	Parâmetro não utilizado
<b>cMensagem</b>	Parâmetro não utilizado
<b>cTudoOk</b>	Expressão para validação da Enchoice
<b>oWnd</b>	Objeto (janela, painel, etc.) onde a enchoice será criada.
<b>IF3</b>	Indica se a enchoice esta sendo criada em uma consulta F3 para utilizar variáveis de memória
<b>IMemoria</b>	Indica se a enchoice utilizará variáveis de memória ou os campos da tabela na edição
<b>IColumn</b>	Indica se a apresentação dos campos será em forma de coluna
<b>caTela</b>	Nome da variável tipo "private" que a enchoice utilizará no lugar da propriedade aTela
<b>INoFolder</b>	Indica se a enchoice não irá utilizar as Pastas de Cadastro (SXA)
<b>IProperty</b>	Indica se a enchoice não utilizará as variáveis aTela e aGets, somente suas propriedades com os mesmos nomes

## **Exemplo: Utilização da função Enchoice()**

```
#include "protheus.ch"

/*
+-----+
| Função | MBRWENCH | Autor | ARNALDO RAYMUNDO JR.|Data |
+-----+
| Descrição | Programa que demonstra a utilização da função Enchoice() |
+-----+
| Uso | Curso ADVPL |
+-----+
 */

User Function MrbwEnch()

Private cCadastro := " Cadastro de Clientes"
Private aRotina := {{"Pesquisar" , "axPesqui" , 0, 1}, {"Visualizar" , "U_ModEnc" , 0, 2} }

DbSelectArea("SA1")
DbSetOrder(1)

MBrowse(6,1,22,75,"SA1")

Return

User Function ModEnc(cAlias,nReg,nOpc)

Local aCpoEnch := {}
Local aAlter := {}

Local cAliasE := cAlias
Local aAlterEnch := {}
Local aPos := {000,000,400,600}
Local nModelo := 3
Local lF3 := .F.
Local lMemoria := .T.
Local lColumn := .F.
Local caTela := ""
Local lNoFolder := .F.
Local lProperty := .F.
Private oDlg
Private oGetD
Private oEnch
Private aTEL[0][0]
Private aGETS[0]

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek(cAliasE)
```

Exemplo (continuação):

```
While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
    If !(SX3->X3_CAMPO $ "A1_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And. ;
        X3Uso(SX3->X3_USADO)
        AADD(aCpoEnch, SX3->X3_CAMPO)
    EndIf
    DbSkip()
End

aAlterEnch := aClone(aCpoEnch)

DEFINE MSDIALOG oDlg TITLE cCadastro FROM 000,000 TO 400,600 PIXEL
RegToMemory("SA1", If(nOpc==3,.T.,.F.))

    Enchoice(cAliasE, nReg, nOpc, /*aCRA*/, /*cLetra*/, /*cTexto*/, aAcho,
              aCpoEnch, aPos, aAlterEnch, nModelo, /*nColMens*/;;
              /*cMensagem*/, /*cTudoOk*/, oDlg, lF3, lMemoria, lColumn,;
              caTela, lNoFolder, lProperty)

ACTIVATE MSDIALOG oDlg CENTERED

Return
```

### 5.1.2. MsMGet()

- Sintaxe:** **MsMGet():New( cAlias, nReg, nOpc, aCRA, cLetra, cTexto, aAcho, aPos, aCpos, nModelo, nColMens, cMensagem, cTudoOk, oWnd, IF3, IMemoria, IColumn, caTela, INoFolder, IProperty)**
- Retorno:** **oMsMGet → objeto do tipo MsMGet()**
- Parâmetros:**

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

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

### Exemplo: Utilização do objeto MsMGet()

```
#include "protheus.ch"

/*
+-----+
| Função | MBRWMSGET | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Programa que demonstra a utilização do objeto MsMget() |
+-----+
| Uso | Curso ADVPL |
+-----+
 */

User Function MrbwMsGet()

Private cCadastro := " Cadastro de Clientes"
Private aRotina := {{"Pesquisar" , "axPesqui" , 0, 1}, {"Visualizar" , "U_ModEnc" , 0, 2} }

DbSelectArea("SA1")
DbSetOrder(1)

MBrowse(6,1,22,75,"SA1")

Return

User Function ModEnc(cAlias,nReg,nOpc)
Local aCpoEnch := {}
Local aAlter := {}

Local cAliasE := cAlias
Local aAlterEnch := {}
Local aPos := {000,000,400,600}
Local nModelo := 3
Local lF3 := .F.
Local lMemoria := .T.
Local lColumn := .F.
Local caTela := ""
Local lNoFolder := .F.
Local lProperty := .F.

Private oDlg
Private oGetD
Private oEnch
Private aTEL[0][0]
Private aGETS[0]
```

**Exemplo (continuação) :**

```
DbSelectArea("SX3")
DbSetOrder(1)
DbSeek(cAliasE)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
    If !(SX3->X3_CAMPO $ "A1_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And.
    X3Uso(SX3->X3_USADO)
        AADD(aCpoEnch, SX3->X3_CAMPO)
    EndIf
    DbSkip()
End

aAlterEnch := aClone(aCpoEnch)

oDlg := MSDIALOG():New(000,000,400,600,cCadastro,,,,,,,,,T..)

RegToMemory(cAliasE, If(nOpc==3,.T.,.F.))

    oEnch := MsMGet():New(cAliasE, nReg, nOpc, /*aCRA*/, /*cLetra*/;;
    /*cTexto*/, aCpoEnch, aPos, aAlterEnch, nModelo, /*nColMens*/;;
    /*cMensagem*/, /*cTudoOk*/, oDlg, lF3, lMemoria, lColumn, caTela,;
    lNoFolder, lProperty)

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

Return
```

## 5.2. Captura de múltiplas informações (Multi-Lines)

A linguagem ADVPL permite a utilização de basicamente dois tipos de objetos do tipo grid, ou como também são conhecidos: multi-line:

- Grids digitáveis:** permitem a visualização e captura de informações, comumente utilizados em interfaces de cadastro e manutenção, tais como:

- MSGETDB()**
- MSGETDADOS()**
- MSNEWGETDADOS()**

- Grids não digitáveis:** permitem somente a visualização de informações, comumente utilizados como browses do ERP Protheus, tais como:

- TWBROWSE()**
- MAWNDBROWSE()**
- MBROWSE()**

Neste tópico serão tratadas as grids digitáveis disponíveis na linguagem ADVPL para o desenvolvimento de interfaces de cadastros e manutenção de informações.

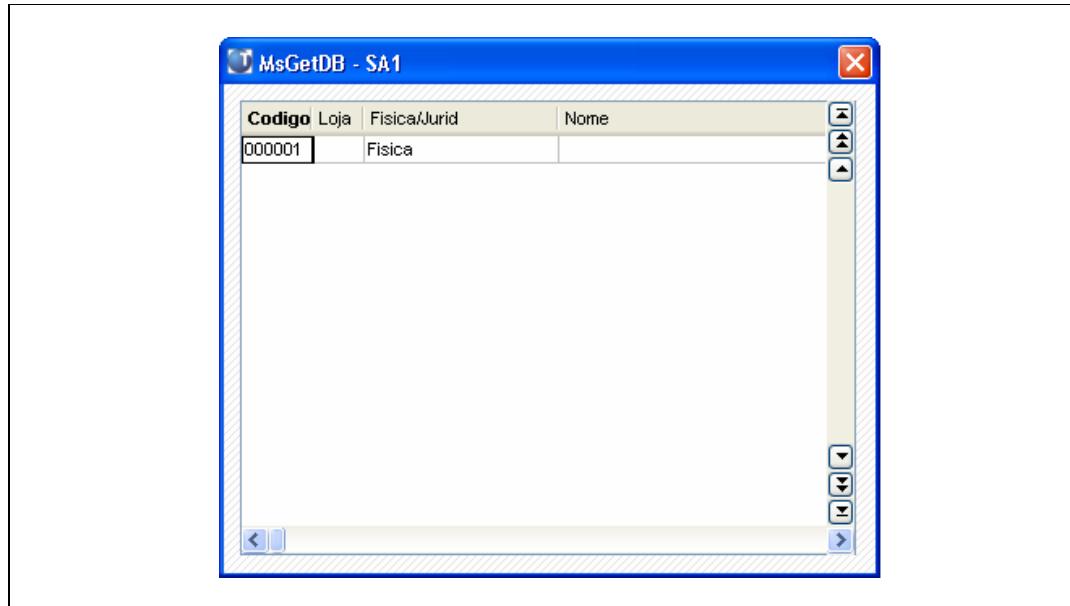
### 5.2.1. MsGetDB()

A classe de objetos visuais MsGetDB() permite a criação de um grid digitável com uma ou mais colunas, baseado em uma tabela temporária.

- Sintaxe:** `MsGetDB():New(nTop, nLeft, nBottom, nRight, nOpc, cLinhaOk, cTudoOk, cIniCpos, IDelete, aAlter, nFreeze, IEmpty, uPar1, cTRB, cFieldOk, ICondicional, IAppend, oWnd, IDisparos, uPar2, cDelOk, cSuperDel)`
- Retorno:** `oMsGetDB` → objeto do tipo MsGetDB()
- Parâmetros:**

<b>nTop</b>	Distancia entre a MsGetDB e o extremidade superior do objeto que a contém.
<b>nLeft</b>	Distancia entre a MsGetDB e o extremidade esquerda do objeto que a contém.
<b>nBottom</b>	Distancia entre a MsGetDB e o extremidade inferior do objeto que a contém.
<b>nRight</b>	Distancia entre a MsGetDB e o extremidade direita do objeto que a contém.
<b>nOpc</b>	Posição do elemento do vetor aRotina que a MsGetDB usará como referência.
<b>cLinhaOk</b>	Função executada para validar o contexto da linha atual do aCols.
<b>cTudoOk</b>	Função executada para validar o contexto geral da MsGetDB (todo aCols).
<b>cIniCpos</b>	Nome dos campos do tipo caracter que utilizarão incremento automático. Este parâmetro deve ser no formato "+<nome do primeiro campo>+<nome do segundo campo>+...".
<b>IDelete</b>	Habilita a opção de excluir linhas do aCols. Valor padrão falso.
<b>aAlter</b>	Vetor com os campos que poderão ser alterados.
<b>nFreeze</b>	Indica qual coluna não ficara congelada na exibição.
<b>IEmpty</b>	Habilita validação da primeira coluna do aCols para esta não poder estar vazia. Valor padrão falso.
<b>uPar1</b>	Parâmetro reservado.
<b>cFieldOk</b>	Função executada na validação do campo.
<b>cTRB</b>	Alias da tabela temporária.
<b>ICondicional</b>	Reservado
<b>IAppend</b>	Indica se a MsGetDB ira criar uma linha em branco automaticamente quando for inclusão.
<b>cDelOk</b>	Função executada para validar a exclusão de uma linha do aCols.
<b>IDisparos</b>	Indica se será utilizado o Dicionário de Dados para consulta padrão, inicialização padrão e gatilhos.
<b>uPar2</b>	Parâmetro reservado.
<b>cSuperDel</b>	-Função executada quando pressionada as teclas <Ctrl>+<Delete>.
<b>oWnd</b>	Objeto no qual a MsGetDB será criada.

**Aparência:**



**Variáveis private:**

<b>aRotina</b>	Vetor com as rotinas que serão executadas na MBrowse e que definirão o tipo de operação que está sendo executada (inclusão, alteração, exclusão, visualização, pesquisa, ...) no formato:  {cTítulo, cRotina, nOpção, nAcesso}, onde:  nOpção segue o padrão do ERP Protheus para:  1- Pesquisar 2- Visualizar 3- Incluir 4- Alterar 5- Excluir
<b>aHeader</b>	Vetor com informações das colunas no formato:  {cTítulo, 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 MsGetDB com a tabela e posiciona na primeira linha.
-----------------------	--

**Exemplo: Utilização do objeto MsGetDB()**

```
#include "protheus.ch"

/*
+-----+
| Função | GETDBSA1      | Autor | MICROSIGA           | Data   |
+-----+
| Descrição | Programa que demonstra a utilização do objeto MsGetDB() |
+-----+
| Uso       | Curso ADVPL          |
+-----+
*/

User Function GetDbSA1()

Local nI
Local oDlg
Local oGetDB
Local nUsado := 0
Local aStruct := {}

Private lRefresh := .T.
Private aHeader := {}
Private aCols := {}
Private aRotina := { {"Pesquisar", "AxPesqui", 0, 1},;
                   {"Visualizar", "AxVisual", 0, 2},;
                   {"Incluir", "AxInclui", 0, 3},;
                   {"Alterar", "AxAltera", 0, 4},;
                   {"Excluir", "AxDelete", 0, 5} }

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek("SA1")
```

Exemplo (continuação):

```
While !Eof() .and. SX3->X3_ARQUIVO == "SA1"
If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL
    nUsado++
    AADD(aHeader, {Trim(X3Titulo()),;
                    SX3->X3_CAMPO,;
                    SX3->X3_PICTURE,;
                    SX3->X3_TAMANHO,;
                    SX3->X3_DECIMAL,;
                    SX3->X3_VALID,;
                    "",;
                    SX3->X3_TIPO,;
                    "",;
                    "" })
    AADD(aStruct, {SX3->X3_CAMPO, SX3->X3_TIPO, SX3->X3_TAMANHO,;
                    SX3->X3_DECIMAL})

EndIf
DbSkip()
End

AADD(aStruct, {"FLAG", "L", 1, 0})

cCriaTrab := CriaTrab(aStruct,.T.)
DbUseArea(.T.,__LocalDriver,cCriaTrab,,.T.,.F.)

oDlg := MSDIALOG():New(000,000,300,400, "MsGetDB - SA1", , , , , .T.)

oGetDB := MsGetDB():New(05,05,145,195,3,"U_LINHAOK", "U_TUDOOK", "+A1_COD", ;
.T., {"A1_NOME"}, 1,.F.,,cCriaTrab,"U_FIELDOK",,.T.,oDlg, .T., , "U_DELOK",;
"U_SUPERDEL")

oDlg:lCentered := .T.
oDlg:Activate()
DbSelectArea(cCriaTrab)
DbCloseArea()

Return

User Function LINHAOK()
ApMsgStop("LINHAOK")
Return .T.

User Function TUDOOK()
ApMsgStop("LINHAOK")
Return .T.

User Function DELOK()
ApMsgStop("DELOK")
Return .T.

User Function SUPERDEL()
ApMsgStop("SUPERDEL")
Return .T.

User Function FIELDOK()
ApMsgStop("FIELDOK")
Return .T.
```

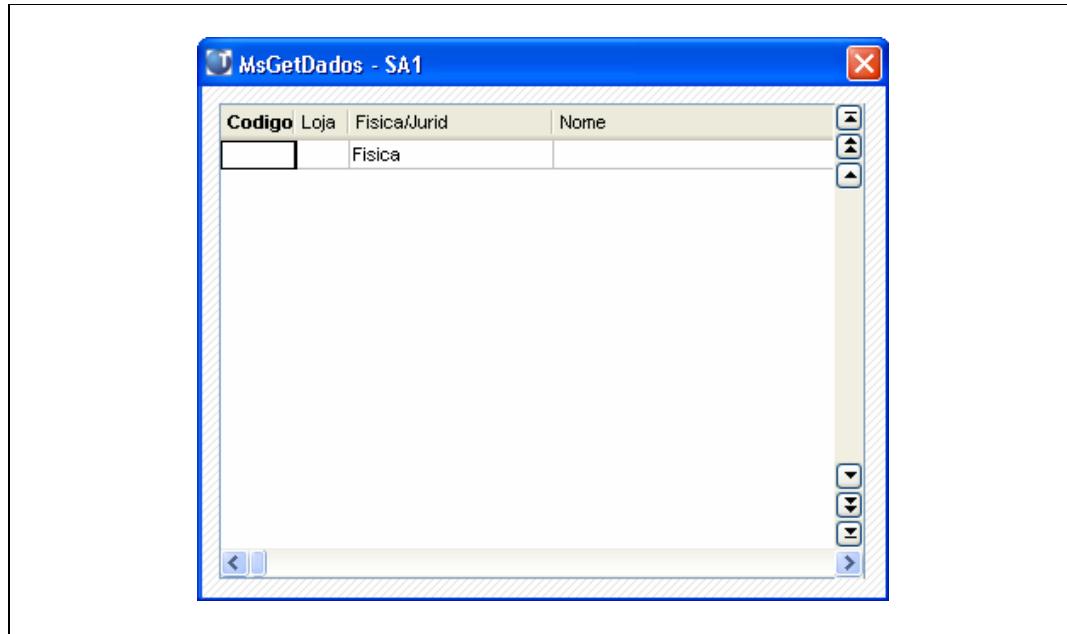
## 5.2.2. MsGetDados()

A classe de objetos visuais MsGetDados() permite a criação de um grid digitável com uma ou mais colunas, baseado em um array.

- Sintaxe:** `MsGetDados():New( nTop, nLeft, nBottom, nRight, nOpc, cLinhaOk, cTudoOk, cIniCpos, IDelete, aAlter, uPar1, IEmpty, nMax, cFieldOk, cSuperDel, uPar2, cDelOk, oWnd)`
- Retorno:** `oMsGetDados → objeto do tipo MsGetDados()`
- Parâmetros:**

<b>nTop</b>	Distancia entre a MsGetDados e o extremidade superior do objeto que a contém.
<b>nLeft</b>	Distancia entre a MsGetDados e o extremidade esquerda do objeto que a contém.
<b>nBottom</b>	Distancia entre a MsGetDados e o extremidade inferior do objeto que a contém.
<b>nRight</b>	Distancia entre a MsGetDados e o extremidade direita do objeto que a contém.
<b>nOpc</b>	Posição do elemento do vetor aRotina que a MsGetDados usará como referencia.
<b>cLinhaOk</b>	Função executada para validar o contexto da linha atual do aCols.
<b>cTudoOk</b>	Função executada para validar o contexto geral da MsGetDados (todo aCols).
<b>cIniCpos</b>	Nome dos campos do tipo caracter que utilizarão incremento automático. Este parâmetro deve ser no formato "+<nome do primeiro campo>+<nome do segundo campo>+...".
<b>IDelete</b>	Habilita excluir linhas do aCols. Valor padrão falso.
<b>aAlter</b>	Vetor com os campos que poderão ser alterados.
<b>uPar1</b>	Parâmetro reservado.
<b>IEmpty</b>	Habilita validação da primeira coluna do aCols para esta não poder estar vazia. Valor padrão falso.
<b>nMax</b>	Número máximo de linhas permitidas. Valor padrão 99.
<b>cFieldOk</b>	Função executada na validação do campo.
<b>cSuperDel</b>	Função executada quando pressionada as teclas <Ctrl> +<Delete>.
<b>uPar2</b>	Parâmetro reservado.
<b>cDelOk</b>	Função executada para validar a exclusão de uma linha do aCols.
<b>oWnd</b>	Objeto no qual a MsGetDados será criada.

**Aparência:**



**Variáveis private:**

<b>aRotina</b>	Vetor com as rotinas que serão executadas na MBrowse e que definirá o tipo de operação que está sendo executada (inclusão, alteração, exclusão, visualização, pesquisa, ...) no formato:  {cTitulo, cRotina, nOpção, nAcesso}, aonde:  nOpção segue o padrão do ERP Protheus para:  6- Pesquisar 7- Visualizar 8- Incluir 9- Alterar 10-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>	Oculta a MsGetDados.
<b>Show()</b>	Mostra a MsGetDados.

**Exemplo: Utilização do objeto MsGetDados()**

```
#include "protheus.ch"

/*
+-----+
| Função | GETDADOSA1 | Autor | MICROSIGA           | Data   |
+-----+
| Descrição | Programa que demonstra a utilização do objeto MSGETADOS() |
+-----+
| Uso      | Curso ADVPL          |
+-----+
*/

User Function GetDadoSA1()

Local 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", "AxDelete", 0, 5}};

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek("SA1")
```

Exemplo (continuação):

```
While !Eof() .and. SX3->X3_ARQUIVO == "SA1"
If X3Uso(SX3->X3_USADO) .and. cNivel >= SX3->X3_NIVEL
    nUsado++
    AADD(aHeader, {Trim(X3Titulo()),;
                    SX3->X3_CAMPO,;
                    SX3->X3_PICTURE,;
                    SX3->X3_TAMANHO,;
                    SX3->X3_DECIMAL,;
                    SX3->X3_VALID,;
                    "",;
                    SX3->X3_TIPO,;
                    "",;
                    "" })
EndIf
DbSkip()
End

AADD(aCols,Array(nUsado+1))

For 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, "U_LINHAOK", "U_TUDOOK",;
    "+A1_COD", .T., {"A1_NOME"}, , .F., 200, "U_FIELDOK", "U_SUPERDEL",;
    "U_DELOK", oDlg)

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

Return

User Function LINHAOK()
ApMsgStop("LINHAOK")
Return .T.

User Function TUDOOK()
ApMsgStop("LINHAOK")
Return .T.

User Function DELOK()
ApMsgStop("DELOK")
Return .T.

User Function SUPERDEL()
ApMsgStop("SUPERDEL")
Return .T.

User Function FIELDOK()
ApMsgStop("FIELDOK")
Return .T.
```

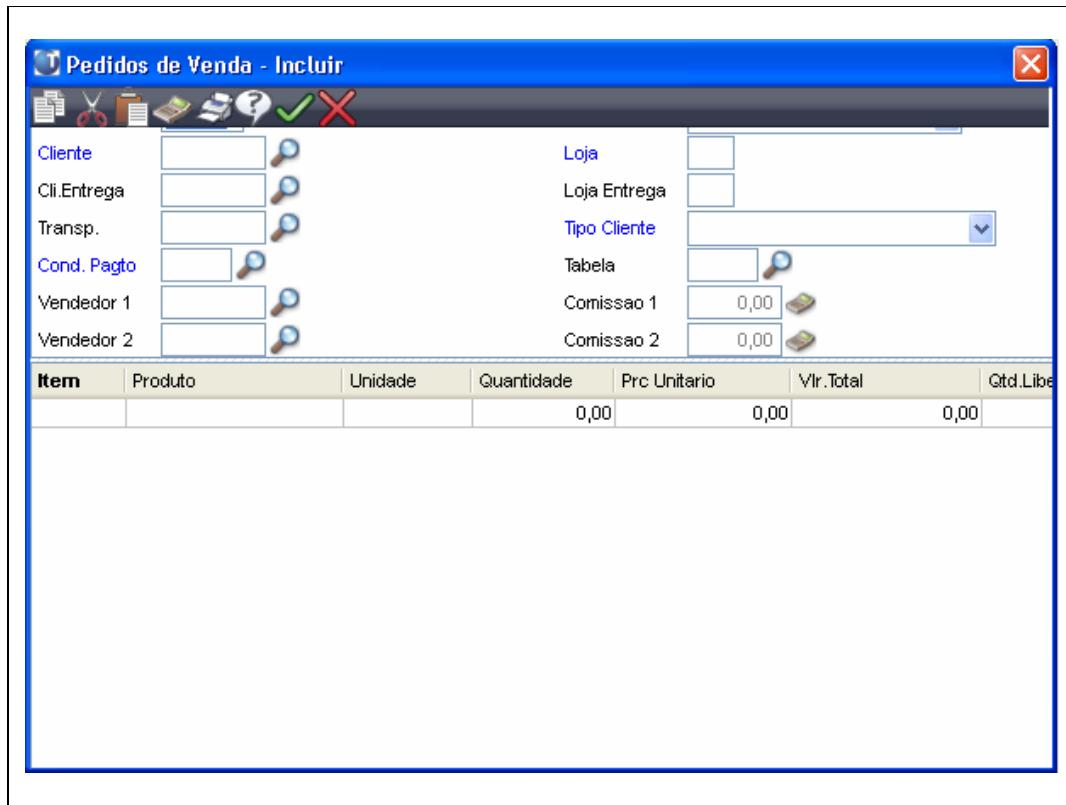
### 5.2.3. MsNewGetDados()

A classe de objetos visuais MsNewGetDados() permite a criação de um grid digitável com uma ou mais colunas, baseado em um array.

- Sintaxe:** `MsNewGetDados():New(nSuperior, nEsquerda ,nInferior, nDireita, nOpc, cLinOk, cTudoOk, cIniCpos, aAlterGDa, nFreeze, nMax, cFieldOk, cSuperDel, cDelOk, oDLG, aHeader, aCols)`
- Retorno:** `oMsGetDados → objeto do tipo MsNewGetDados()`
- Parâmetros:**

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

**Aparência:**



**Variáveis private:**

<b>aRotina</b>	<p>Vetor com as rotinas que serão executadas na MBrowse e que definira o tipo de operação que esta sendo executada (inclusão, alteração, exclusão, visualização, pesquisa, ...) no formato:  <code>{cTitulo, cRotina, nOpção, nAcesso}</code>, aonde:  nOpção segue o padrão do ERP Protheus para:</p> <ul style="list-style-type: none"> <li>1- Pesquisar</li> <li>2- Visualizar</li> <li>3- Incluir</li> <li>4- Alterar</li> <li>5- Excluir</li> </ul>
<b>aHeader</b>	<p>Vetor com informações das colunas no formato:  <code>{cTitulo, cCampo, cPicture, nTamanho, nDecimais,; cValidação, cReservado, cTipo, xReservado1, xReservado2}</code></p> <p>A tabela temporária utilizada pela MsGetDB deverá ser criada com base no aHeader mais um último campo tipo lógico que determina se a linha foi excluída.</p>
<b>IRefresh</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 MsNewGetDados com a tabela e posiciona na primeira linha.
<b>Hide()</b>	Oculta a MsNewGetDados.
<b>Show()</b>	Mostra a MsNewGetDados.

**Exemplo: Utilização dos objetos MsNewGetDados() e MsMGet()**

```
#include "protheus.ch"

/*
+-----+
| Função | MBRWGETD | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Programa que demonstra a utilização dos objetos
|           | MsNewGetDados() e MsMGet() combinados
+-----+
| Uso      | Curso ADVPL
+-----+
*/

User Function MrbwGetD()

Private cCadastro := "Pedidos de Venda"
Private aRotina   :=   {{"Pesquisar" , "axPesqui" , 0, 1},;
                      {"Visualizar" , "U_ModGtd" , 0, 2},;
                      {"Incluir"    , "U_ModGtd" , 0, 3}};

DbSelectArea("SC5")
DbSetOrder(1)

MBrowse(6,1,22,75,"SC5")

Return

User Function ModGtd(cAlias,nReg,nOpc)

Local nX          := 0
Local nUsado       := 0
Local aButtons     := {}
Local aCpoEnch    := {}
Local cAliasE      := cAlias
Local aAlterEnch   := {}
```

Exemplo (continuação):

```
Local aPos      := {000,000,080,400}
Local nModelo   := 3
Local lF3       := .F.
Local lMemoria  := .T.
Local lColumn    := .F.
Local caTela     := ""
Local lNoFolder  := .F.
Local lProperty  := .F.
Local aCpoGDa   := {}
Local cAliasGD  := "SC6"
Local nSuperior  := 081
Local nEsquerda  := 000
Local nInferior  := 250
Local nDireita   := 400
Local cLinOk     := "AllwaysTrue"
Local cTudoOk    := "AllwaysTrue"
Local cIniCpos   := "C6_ITEM"
Local nFreeze    := 000
Local nMax       := 999
Local cFieldOk   := "AllwaysTrue"
Local cSuperDel  := ""
Local cDelOk     := "AllwaysFalse"
Local aHeader    := {}
Local aCols      := {}
Local aAlterGDa  := {}

Private oDlg
Private oGetD
Private oEnch
Private aTEL A[0][0]
Private aGETS[0]

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek(cAliasE)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
  If !(SX3->X3_CAMPO $ "C5_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And. ;
  X3Uso(SX3->X3_USADO)
    AADD(aCpoEnch, SX3->X3_CAMPO)
  EndIf
  DbSkip()
End

aAlterEnch := aClone(aCpoEnch)

DbSelectArea("SX3")
DbSetOrder(1)
MsSeek(cAliasGD)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasGD
  If !(AllTrim(SX3->X3_CAMPO) $ "C6_FILIAL") .And. ;
  cNivel >= SX3->X3_NIVEL .And. X3Uso(SX3->X3_USADO)
    AADD(aCpoGDa, SX3->X3_CAMPO)
  EndIf
  DbSkip()
End
```

Exemplo (continuação):

```
aAlterGDa := aClone(aCpoGDa)

nUsado:=0
dbSelectArea("SX3")
dbSeek("SC6")
aHeader:={}
While !Eof().And.(x3_arquivo=="SC6")
    If X3USO(x3_usado).And.cNivel>=x3_nivel
        nUsado:=nUsado+1
    AADD(aHeader,{ TRIM(x3_titulo), x3_campo, x3_picture,x3_tamanho,;
    x3_decimal,"AlwaysTrue()",x3_usado, x3_tipo, x3_arquivo, x3_context } )
        Endif
    dbSkip()
End

If nOpc==3 // Incluir
    aCols:={Array(nUsado+1)}
    aCols[1,nUsado+1]:=.F.
    For nX:=1 to nUsado
        IF aHeader[nX,2] == "C6_ITEM"
            aCols[1,nX]:= "0001"
        ELSE
            aCols[1,nX]:=CriaVar(aHeader[nX,2])
        ENDIF
    Next
Else
    aCols:={}
    dbSelectArea("SC6")
    dbSetOrder(1)
    dbSeek(xFilial() +M->C5_NUM)
    While !eof().and.C6_NUM==M->C5_NUM
        AADD(aCols,Array(nUsado+1))
        For nX:=1 to nUsado
            aCols[Len(aCols),nX]:=FieldGet(FieldPos(aHeader[nX,2]))
        Next
        aCols[Len(aCols),nUsado+1]:=.F.
        dbSkip()
    End
endif

oDlg := MSDIALOG():New(000,000,400,600, cCadastro,,,,,,,,,T.)
RegToMemory("SC5", If(nOpc==3,.T.,.F.))

oEnch := MsMGet():New(cAliasE,nReg,nOpc,/*cAra*/,/*cLetra*/,/*cTexto*/,,;
    aCpoEnch,aPos,aAlterEnch, nModelo, /*nColMens*/,, /*cMensagem*/,,;
    /*cTudoOk*/,, oDlg,lF3, lMemoria,lColumn,caTela,lNoFolder,;
    lProperty)

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

oDlg:bInit := {|| EnchoiceBar(oDlg, {||oDlg:End()}, {||oDlg:End()},,aButtons)}
oDlg:lCentered := .T.
oDlg:Activate()
Return
```

### **5.2.3.1. Definindo cores personalizadas para o objeto MsNewGetDados()**

---

Conforme visto no tópico sobre definição das propriedades de cores para os componentes visuais, cada objeto possui características que devem ser respeitadas para correta utilização deste recurso.

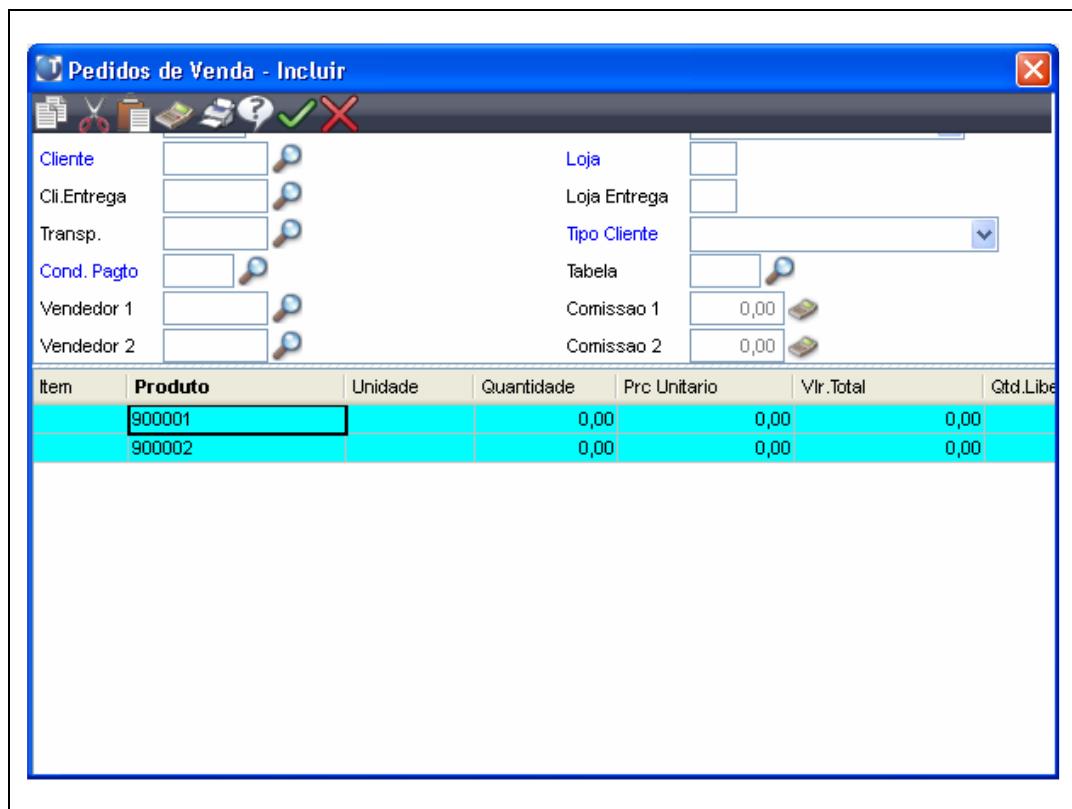
**Atributos adicionais:**

<b>IUseDefaultColors</b>	Atributo que deverá ser definido como .F. para que as alterações nas cores sejam permitidas.
--------------------------	--

**Métodos adicionais:**

<b>SetBkColor</b>	Método que define a cor que será utilizada para cada linha do grid. Não é necessário utilizar o método Refresh() após a definição da cor por este método.
-------------------	---

**Aparência:**



## **Exemplo: Definindo cores personalizadas para o objeto MsNewGetDados()**

```
#include "protheus.ch"

/*
+-----+
| Função | MRBWGTCL | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Programa que demonstra a utilização dos objetos |
|           | MsNewGetDados() e MsMGet() combinados e tratamento de cores |
+-----+
| Uso      | Curso ADVPL |
+-----+
 */

User Function MrbwGtCl()

Private cCadastro := "Pedidos de Venda"
Private aRotina := {{"Pesquisar" , "axPesqui" , 0, 1},;
                    {"Visualizar" , "U_ModGtd" , 0, 2},;
                    {"Incluir" , "U_ModGtd" , 0, 3}};

DbSelectArea("SC5")
DbSetOrder(1)

MBrowse(6,1,22,75,"SC5")

Return

User Function ModGtd(cAlias,nReg,nOpc)

Local nX      := 0
Local nUsado   := 0

Local aButtons  := {}
Local aCpoEnch  := {}
Local cAliasE   := cAlias
Local aAlterEnch := {}
Local aPos       := {000,000,080,400}
Local nModelo    := 3
Local lF3        := .F.
Local lMemoria   := .T.
Local lColumn    := .F.
Local caTela     := ""
Local lNoFolder  := .F.
Local lProperty   := .F.
Local aCpoGDa   := {}
Local cAliasGD   := "SC6"
Local nSuperior  := 081
Local nEsquerda  := 000
Local nInferior   := 250
Local nDireita    := 400
Local cLinOk     := "AlwaysTrue"
Local cTudoOk    := "AlwaysTrue"
Local cIniCpos   := "C6_ITEM"
Local nFreeze    := 000
Local nMax       := 999
```

Exemplo (continuação):

```
Local cFieldOk          := "AllwaysTrue"
Local cSuperDel         := ""
Local cDelOk            := "AllwaysFalse"
Local aHeader           := {}
Local aCols             := {}
Local aAlterGDa         := {}

Private oDlg
Private oGetD
Private oEnch
Private aTEL A[0][0]
Private aGETS[0]

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek(cAliasE)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasE
    If !(X3->X3_CAMPO $ "C5_FILIAL") .And. cNivel >= SX3->X3_NIVEL .And. ;
    X3Uso(SX3->X3_USADO)
        AADD(aCpoEnch, SX3->X3_CAMPO)
    EndIf
    DbSkip()
End

aAlterEnch := aClone(aCpoEnch)

DbSelectArea("SX3")
DbSetOrder(1)
MsSeek(cAliasGD)

While !Eof() .And. SX3->X3_ARQUIVO == cAliasGD
    If !(AllTrim(SX3->X3_CAMPO) $ "C6_FILIAL") .And. cNivel >= SX3-
    >X3_NIVEL .And. X3Uso(SX3->X3_USADO)
        AADD(aCpoGDa, SX3->X3_CAMPO)
    EndIf
    DbSkip()
End

aAlterGDa := aClone(aCpoGDa)

nUsado:=0
dbSelectArea("SX3")
dbSeek("SC6")
aHeader:={}
While !Eof().And.(x3_arquivo=="SC6")
    If X3USO(x3_usado).And.cNivel>=x3_nivel
        nUsado:=nUsado+1
    AADD(aHeader, { TRIM(x3_titulo), x3_campo, x3_picture,;
                    x3_tamanho, x3_decimal,"AllwaysTrue()",;
                    x3_usado, x3_tipo, x3_arquivo, x3_context } )
    Endif
    dbSkip()
End
```

Exemplo (continuação):

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

        IF aHeader[nX,2] == "C6_ITEM"
            aCols[1,nX]:= "0001"
        ELSE
            aCols[1,nX]:=CriaVar(aHeader[nX,2])
        ENDIF

    Next
Else
    aCols:={}
    dbSelectArea("SC6")
    dbSetOrder(1)
    dbSeek(xFilial()+M->C5_NUM)
    While !eof().and.C6_NUM==M->C5_NUM
        AADD(aCols,Array(nUsado+1))
        For nX:=1 to nUsado
            aCols[Len(aCols),nX]:=FieldGet(FieldPos(aHeader[nX,2]))
        Next
        aCols[Len(aCols),nUsado+1]:=F.
        dbSkip()
    End
Endif

oDlg := MSDIALOG():New(000,000,400,600, cCadastro,,,.T..)

    RegToMemory("SC5", If(nOpc==3,.T.,.F.))

    oEnch := MsMGet():New(cAliasE,nReg,nOpc,/*aCRA*/,/*cLetra*/, /*cTexto*/;,;
                           aCpoEnch,aPos, aAlterEnch, nModelo, /*nColMens*/, /*cMensagem*/;,;
                           cTudoOk,oDlg,1F3, lMemoria,lColumn,caTela,lNoFolder,lProperty)

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

    // Tratamento para definição de cores específicas,
    // logo após a declaração da MsNewGetDados

    oGetD:oBrowse:lUseDefaultColors := .F.
    oGetD:oBrowse:SetBlkBackColor({|| GETDCLR(oGetD:aCols,oGetD:nAt,aHeader)})

oDlg:bInit := {|| EnchoiceBar(oDlg, {||oDlg:End()}, {||oDlg:End()},,aButtons)}
oDlg:lCentered := .T.
oDlg:Activate()

Return
```

Exemplo (continuação):

```
// Função para tratamento das regras de cores para a grid da MsNewGetDados

Static Function GETDCLR(aLinha,nLinha,aHeader)

Local nCor2      := 16776960 // Ciano - RGB(0,255,255)
Local nCor3      := 16777215 // Branco - RGB(255,255,255)
Local nPosProd   := aScan(aHeader,{|x| Alltrim(x[2]) == "C6_PRODUTO"})
Local nUsado     := Len(aHeader)+1
Local nRet       := nCor3

If !Empty(aLinha[nLinha][nPosProd]) .AND. aLinha[nLinha][nUsado]
    nRet := nCor2
ElseIf !Empty(aLinha[nLinha][nPosProd]) .AND. !aLinha[nLinha][nUsado]
    nRet := nCor3
Endif

Return nRet
```



Anotações

---

---

---

---

## 5.3. Barras de botões

A linguagem ADVPL permite a implementação de barras de botões utilizando funções pré-definidas desenvolvidas com o objetivo de facilitar sua utilização, ou através da utilização direta dos componentes visuais disponíveis. Dentre os recursos da linguagem que podem ser utilizados com esta finalidade serão abordados:

- Função EnchoiceBar:** Sintaxe tradicionalmente utilizada em ADVPL, a qual não retorna um objeto para a aplicação chamadora;
- Classe TBar:** Classe do objeto TBar(), a qual permite a instanciação direta de um objeto do tipo barra de botões superior, tornando-o disponível na aplicação chamadora.
- Classe ButtonBar:** Classe do objeto ButtonBar(), a qual permite a instanciação direta de um objeto barra de botões genérico, o qual pode ser utilizado em qualquer posição da tela, tornando-o disponível na aplicação chamadora.

### 5.3.1. EnchoiceBar()

Função que cria uma barra de botões no formato padrão utilizado pelas interfaces de cadastro da aplicação Protheus.

Esta barra possui os botões padrões para confirmar ou cancelar a interface e ainda permite a adição de botões adicionais com a utilização do parâmetro **aButtons**.

**Sintaxe:**

```
EnchoiceBar( oDlg, bOk, bCancel, IMsgDel, aButtons, nRecno, cAlias)
```

**Parâmetros:**

<b>oDlg</b>	Dialog onde irá criar a barra de botões
<b>bOk</b>	Bloco de código a ser executado no botão Ok
<b>bCancel</b>	Bloco de código a ser executado no botão Cancelar
<b>IMsgDel</b>	Exibe dialog para confirmar a exclusão
<b>aButtons</b>	Array contendo botões adicionais. aArray[n][1] -> Imagem do botão aArray[n][2] -> bloco de código contendo a ação do botão aArray[n][3] -> título do botão
<b>nRecno</b>	Registro a ser posicionado após a execução do botão Ok.
<b>cAlias</b>	Alias do registro a ser posicionado após a execução do botão Ok. Se o parâmetro nRecno for informado, o cAlias passa ser obrigatório.

**Aparência:**



## **Exemplo: Utilização da função EnchoiceBar()**

```
#include "protheus.ch"

/*
+-----+
| Função | DENCHBAR | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Programa que demonstra a utilização da função
|           | EnchoiceBar()
+-----+
| Uso      | Curso ADVPL
+-----+
 */

User Function DEnchBar()
Local oDlg, oBtn
Local aButtons := {}

DEFINE MSDIALOG oDlg TITLE "Teste EnchoiceBar" FROM 000,000 TO 400,600 PIXEL OF;
oMainWnd

AADD( aButtons, {"HISTORIC", {|| TestHist()}, "Histórico...",;
"Histórico",{|| .T.}} )

@ -15,-15 BUTTON oBtn PROMPT "..." SIZE 1,1 PIXEL OF oDlg

ACTIVATE MSDIALOG oDlg ;
ON INIT (EnchoiceBar(oDlg,{||Ok:=.T.,oDlg:End()},{{||oDlg:End()}},@aButtons))

Return
```



*Anotações*

---

---

---

---

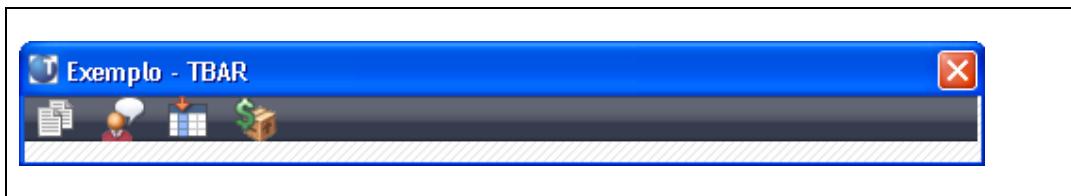
### 5.3.2. TBar()

Classe de objetos visuais que permite a implementação de um componente do tipo barra de botões para a parte superior de uma janela previamente definida.

- Sintaxe:** `New(oWnd, nBtnWidth, nBtnHeight, l3D, cMode, oCursor, cResource, INoAutoAdjust)`
- Retorno:** `oTBar → objeto do tipo TBar()`
- Parâmetros:**

<b>oWnd</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>nBtnWidth</b>	Numérico, opcional. Largura do botão contido na barra
<b>nBtnHeight</b>	Numérico, opcional. Altura do botão contido na barra
<b>l3D</b>	Lógico, opcional. Define tipo da barra
<b>cMode</b>	Não utilizado.
<b>oCursor</b>	Objeto, opcional. Define Cursor ao posicionar o mouse sobre a barra.
<b>cResource</b>	Caracter, opcional. Imagem do recurso a ser inserido como fundo da barra.
<b>INoAutoAdjust</b>	Lógico.

- Aparência:**



#### **Exemplo: Utilização da função EnchoiceBar()**

```
#include 'protheus.ch'

/*
+-----+
| Função | TSTBAR           | Autor | MICROSIGA          | Data |
+-----+
| Descrição | Programa que demonstra a utilização do objeto TBar() |
+-----+
| Uso       | Curso ADVPL          |
+-----+
*/
User Function TstTBar()
Local oDlg

oDlg := MSDIALOG():New(000,000,305,505, 'Exemplo - TBAR', . . . . .T.)
```

**Exemplo (continuação):**

```
oTBar := TBar():New( oDlg,25,32,.T.,,,.F. )

oTBtnBmp2_1 := TBtnBmp2():New( 00, 00, 35, 25, 'copyuser' ,,,,;
{|||Alert('TBtnBmp2_1')}, oTBar,'msGetEx',,.F.,.F. )

oTBtnBmp2_2 := TBtnBmp2():New( 00, 00, 35, 25, 'critica' ,,,,;
{|||},oTBar,'Critica',,.F.,.F. )

oTBtnBmp2_3 := TBtnBmp2():New( 00, 00, 35, 25, 'bmpcpo' ,,,,;
{|||},oTBar,'PCO',,.F.,.F. )

oTBtnBmp2_4 := TBtnBmp2():New( 00, 00, 35, 25, 'preco' ,,,,;
{|||},oTBar,'Preço' ,,.F.,.F. )

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

Return
```

### 5.3.3. ButtonBar

A sintaxe **ButtonBar** é a forma clássica utilizada na linguagem ADVPL para implementar um objeto da classe TBar(), o qual possui as características mencionadas no tópico anterior.

**Sintaxe:**

```
DEFINE BUTTONBAR oBar SIZE nWidth, nHeight 3D MODE OF oDlg
CURSOR
```

**Retorno: ()**.

**Parâmetros:**

<b>oBar</b>	Objeto do tipo TBar() que será criado com a utilização da sintaxe ButtonBar().
<b>nWidth</b>	Numérico, opcional. Largura do botão contido na barra.
<b>nHeight</b>	Numérico, opcional. Altura do botão contido na barra.
<b>3D</b>	Se definido habilita a visualização em 3D da barra de botões.
<b>oDlg</b>	Objeto, opcional. Janela ou controle onde o botão deverá ser criado.
<b>MODE</b>	Define a forma de orientação do objeto ButtonBar utilizando os seguintes termos pré-definidos:  TOP, BOTTOM, FLOAT
<b>CURSOR</b>	Objeto, opcional. Define Cursor ao posicionar o mouse sobre a barra.

A sintaxe ButtonBar requer a adição dos botões como recursos adicionais da barra previamente definida utilizando a sintaxe abaixo:

**Botões: BUTTON RESOURCE**

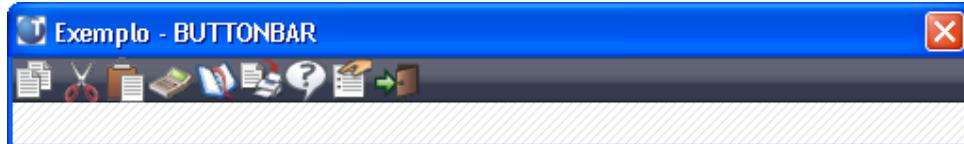
**Sintaxe adicional:**

```
DEFINE BUTTON RESOURCE cBitMap OF oBar ACTION cAcao TOOLTIP  
cTexto
```

**Parâmetros:**

<b>cBitMap</b>	Nome da imagem disponível na aplicação.
<b>oBar</b>	Objeto do tipo TBar() no qual o botão será adicionado.
<b>cAcao</b>	Função ou lista de expressões que determina a ação que será realizada pelo botão.
<b>cTexto</b>	Texto no estilo "tooltip text" que será exibido quando o cursor do mouse for posicionado sobre o botão na barra de ferramentas.

**Aparência:**



### **Exemplo: Utilização da sintaxe ButtonBar**

```
#include 'protheus.ch'

/*
+-----+
| Função | TstBBar           | Autor | MICROSIGA          | Data | |
+-----+
| Descrição | Programa que demonstra a utilização do objeto TBar() | |
+-----+
| Uso      | Curso ADVPL          |          | |
+-----+
*/

User Function TstBBar()

Local oDlg
Local oBtn1
Local oBtn2

oDlg := MSDIALOG():New(000,000,305,505, 'Exemplo - BUTTONBAR', .T.)

DEFINE BUTTONBAR oBar SIZE 25,25 3D TOP OF oDlg
```

**Exemplo (continuação) :**

```
DEFINE BUTTON RESOURCE "S4WB005N" OF oBar ACTION NaoDisp() TOOLTIP "Recortar"
DEFINE BUTTON RESOURCE "S4WB006N" OF oBar ACTION NaoDisp() TOOLTIP "Copiar"
DEFINE BUTTON RESOURCE "S4WB007N" OF oBar ACTION NaoDisp() TOOLTIP "Colar"
DEFINE BUTTON oBtn1 RESOURCE "S4WB008N" OF oBar GROUP;
ACTION Calculadora() TOOLTIP "Calculadora"

oBtn1:cTitle:="Calc"
DEFINE BUTTON RESOURCE "S4WB009N" OF oBar ACTION Agenda() TOOLTIP "Agenda"
DEFINE BUTTON RESOURCE "S4WB010N" OF oBar ACTION OurSpool() TOOLTIP "Spool"
DEFINE BUTTON RESOURCE "S4WB016N" OF oBar GROUP;
ACTION HelProg() TOOLTIP "Ajuda"

DEFINE BUTTON oBtn2 RESOURCE "PARAMETROS" OF oBar GROUP;
ACTION Sx1C020() TOOLTIP "Parâmetros"

oBtn2:cTitle:="Param."

DEFINE BUTTON oBtOk RESOURCE "FINAL" OF oBar GROUP;
ACTION oDlg:End() TOOLTIP "Sair"

oBar:bRClicked := { || AlwaysTrue() }
oDlg:lCentered := .T.
oDlg:Activate()

Return
```



**Anotações**

---

---

---

---

### 5.3.4. Imagens pré-definidas para as barras de botões

Conforme mencionado nos tópicos anteriores, os botões visuais do tipo barra de botões permitem a definição de itens com ações e imagens vinculadas.

Dentre os objetos e funções mencionados, foi citada a EnchoiceBar(), a qual permite a adição de botões adicionais através do parâmetro **aButton**, sendo que os itens deste array devem possuir o seguinte formato:

- Sintaxe:** AADD(aButtons,{cBitMap, bAcao, cTexto})

- Estrutura:**

<b>cBitMap</b>	Nome da imagem pré-definida existente na aplicação ERP que será vinculada ao botão.
<b>bAcao</b>	Bloco de código que define a ação que será executada com a utilização do botão.
<b>cTexto</b>	Texto no estilo “tooltip text” que será exibido quando o cursor do mouse for posicionado sobre o botão na barra de ferramentas.

- Alguns BitMaps disponíveis:**



- Exemplo:**

```
AADD(aButtons, {"USER", {||AlwaysTrue(), "Usuário"})}
```

## 6. Outras aplicações da interface visual do ADVPL

### 6.1. MaWndBrowse()

- Descrição:** Browse que permite a visualização de registros para arquivos / tabelas que não possuem estrutura definida no Dicionário de Dados do sistema.
- Sintaxe:** MaWndBrowse (nLin1, nCol1, nLin2, nCol2, cTitle, cAlias, aCampos, aRotina, cFunLeg, cTopFun, cBotFun, lCentered, aResource, nMod, aPesqui, cSeek, IDic, ISavOrd)
- Parâmetros:**

<b>nLin1</b>	Linha inicial do browse
<b>nCol1</b>	Coluna inicial do browse
<b>nLin2</b>	Linha final do browse
<b>nCol2</b>	Coluna final do browse
<b>cTitle</b>	Título do browse (obrigatório)
<b>cAlias</b>	Alias da tabela corrente podendo ser um temporário
<b>aCampos</b>	Se IDic=.T. utilizará o SX3, do contrário o aCampos informado
<b>aRotina</b>	Idêntico ao aRotina para mBrowse
<b>cFunLeg</b>	Função que deverá retornar um valor lógico e com isso será atribuído semafóro na primeira coluna do browse
<b>cTopFun</b>	Mostrar os registros com a chave de
<b>cBotFun</b>	Mostrar os registros com a chave ate
<b>lCentered</b>	Valor verdadeiro centraliza
<b>aResource</b>	aAdd(aResource, {"IMAGEM", "Texto significativo"})
	Posição do Menu:
<b>nModelo</b>	1- Horizontal (superior) 2- Vertical (esquerda) 3- Horizontal (inferior)
<b>aPesqui</b>	aAdd(aPesqui, {"Título", nOrdem}), se não passado será utilizado o AxPesqui
<b>cSeek</b>	Chave principal para a busca, exemplo: xFilial("??")
<b>IDic</b>	Parâmetro em conjunto com aCampos
<b>ISavOrd</b>	Estabelecer a ordem após pesquisas.

- Estrutura do array aCampos**

aAdd(aCampo, {X3\_CAMPO, X3\_PICTURE, X3\_TITULO, X3\_TAMANHO})

**Aparência:**

Cadastro Temporario					
Pesquisar		Visualizar		Incluir	Alterar
Filial	Matricula	Nome		Idade	Status
01	000001	ARNALDO		30	A
01	000002	ANDREIA		25	A

**Exemplo: MaWndBrowse para arquivo temporário**

```
#include "protheus.ch"

/*
+-----+
| Função      | WndBwTRB      | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição   | Demonstra a utilização da MaWndBrowse com Temporário |      |
+-----+
| Uso         | Curso ADVPL |      |
+-----+
*/

User Function WndBwTRB()

// Variáveis para o Arquivo Temporario
Local cChave      := ""
Local cArqTrb     := ""
Local aStruTRB    := {}

// Variáveis para o MaWndBrowse
Local cTitulo      := "Cadastro Temporario"// Título obrigatório
Local cAlias       := "TRB" // Alias da tabela corrente podendo ser TRB
Local cFunLeg     := "" // Função que deverá retornar um valor lógico e com isso
será atribuído semafóro na primeira coluna do browse
Local cTopFun     := "" // Mostrar os registros com a chave de
Local cBotFun     := "" // Mostrar os registros com a chave ate
Local lCentered   := .T. // Valor verdadeiro centraliza
Local aResource   := {} // aAdd(aResource,"IMAGEM","Texto significativo")
Local nModelo     := 1 // 1- Menu do aRotina
Local aPesqui     := {} // aAdd(aPesqui{"Título",nOrdem}), se não passado será
utilizado o AxPesqui
Local cSeek       := "" // Chave principal para a busca, exemplo: xFilial("??")
Local lDic        := .F. // Parâmetro em conjunto com aCampos
Local lSavOrd     := .T. // Estabelecer a ordem após pesquisas.
```



```

dbSelectArea( "TRB" )
MaWndBrowse(aSize[7],aSize[2],aSize[6],aSize[5],cTitulo,cAlias,aCampos,aRotina,,cTopFun,cBotFun,lCentered,,nModelo,,cSeek,lDic,lSavOrd)

If ( Select( "TRB" ) <> 0 )
    dbSelectArea ( "TRB" )
    dbCloseArea ()
Endif

Return

```

### 6.1.1. Enchoice para Arquivos Temporários

Para aproveitar inteiramente o recurso da MaWndBrowse utilizada com arquivos temporários, é necessário implementar uma função do tipo Enchoice para realizar as operações de Visualizar, Incluir, Alterar e Excluir os registros deste arquivo / tabela.

Com foco nesta necessidade, foi desenvolvida para o curso de ADVPL a função TEnchoice.

- Descrição:** Função para manipulação de registros de um arquivo temporário simular a Enchoice do ambiente Protheus.
- Sintaxe:** TEnchoice(oDlg, aCampos, nLeftE, nTopE, nHeightE, nWidthE, lEnchBar)
- Parâmetros:**

<b>oDlg</b>	Objeto Dialog no qual a TEnchoice será vinculada
<b>aCampos</b>	Estrutura de campos para exibição da tela da TEnchoice
<b>nLeftE</b>	Coordenada horizontal em pixels.
<b>nTopE</b>	Coordenada vertical em pixels.
<b>nHeightE</b>	Altura em pixels
<b>nWidthE</b>	Largura em pixels.
<b>lEnchBar</b>	Se a enchoice será montada em conjunto com uma barra de botões do tipo Enchoicebar()

- Pré-requisitos da função chamadora:**

Para correta utilização da função TEnchoice são necessários os seguintes tratamentos / definições na função chamadora:

- Tratamento 01: Montagem do array aCampos**

O array aCampos utilizado pela TEnchoice possui basicamente o mesmo formato do array utilizado pela função Modelo2().

<b>"&lt;Variavel&gt;"</b>	Nome da variável Private que está vinculada ao campo digitável.
<b>{nLinha,nColuna}</b>	Posição em pixels do campo na Enchoice.
<b>"&lt;Titulo&gt;"</b>	Título do campo.
<b>"&lt;Picture&gt;"</b>	Picture de formatação do campo.
<b>"&lt;Validacao&gt;"</b>	Validação do campo digitável.
<b>"&lt;F3&gt;"</b>	Consulta F3 vinculada ao campo.

<b>"&lt;IWhen&gt;"</b>	Se o campo está editável.
<b>&lt;Tamanho&gt;</b>	Tamanho do campo para visualização.

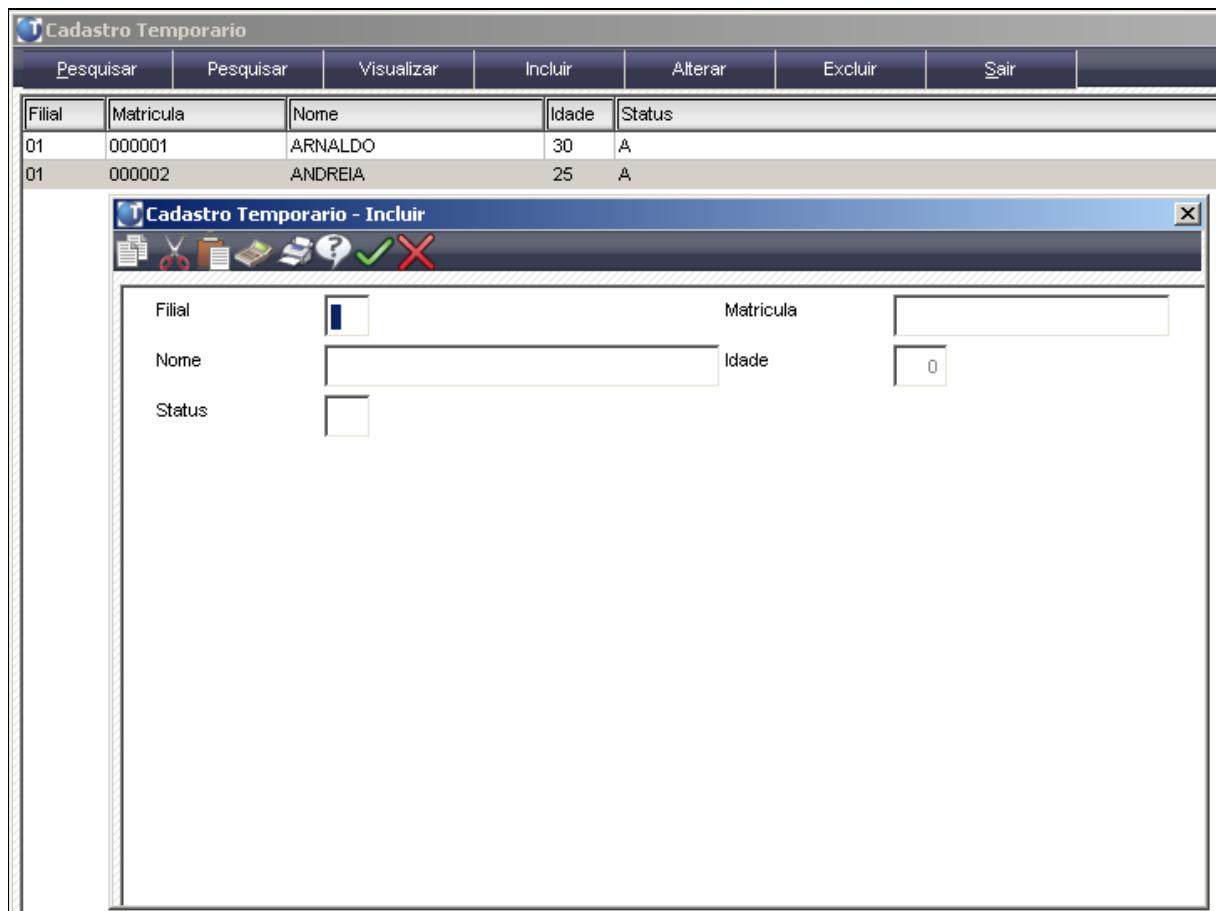
```
AADD(aCposEnch, {"<Variavel>" ,{nLinha,nColuna} ,<Titulo>,";
"<Picture>", "<Validacao>", "<F3>", "<lWhen>,";
<Tamanho>})
```

**Tratamento 02: Definição das dimensões da TEnchoice**

Os parâmetros de dimensão da TEnchoice não precisam ser informados, pois ela irá se ajustar ao tamanho do objeto Dialog ao qual for vinculado, caso o objetivo seja que a TEnchoice ocupe toda a área disponível do objeto.

É importante especificar se o Dialog irá conter uma EnchoiceBar() para que a TEnchoice() seja posicionada adequadamente no objeto.

**Aparência:**



**Detalhamento da função TEnchoice:**

```

/*
+-----+
| Função      | TEnchoice      | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição   | Enchoice para arquivos temporarios |      |
+-----+
| Uso         | Curso ADVPL |      |
+-----+
*/

Static Function TEnchoice(oDlg, aCampos, nLeftE, nTopE, nHeightE, nWidthE,;
lEnchBar)

Local aSays      := {}
Local aGets      := {}
Local cCaption   := ""
Local cPict      := ""
Local cValid     := ""
Local cF3        := ""
Local cWhen      := ""
Local cBlKSay    := ""
Local cBlkGet    := ""
Local cBlKVld   := ""
Local cBlKWhen   := ""
Local nLeft      := 0
Local nTop       := 0
Local nI         := 0

Default lEnchBar := .F.
Default   nLeftE           := IIF(lEnchBar,(oDlg:nLeft)+16,2)
Default   nTopE            := 2
Default   nHeightE         := (oDlg:nHeight)-135
Default nWidthE          := (oDlg:nWidth)-102
// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aC, {"<Variavel>", "", {nLinha,nColuna}
//,"<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>,<Tamanho>})

If Len(aCampos) > 0

    oScroll := TScrollBox():New(oDlg, nLeftE, nTopE, nHeightE , nWidthE, .T.,;
    .T., .T.)

        For nI:=1 to Len(aCampos)

            If Len(aCampos[nI])==8

                cCampo      := aCampos[nI,1]
                nLeft := aCampos[nI,2,1]-13
                nTop  := aCampos[nI,2,2]
                cCaption:= Iif(Empty(aCampos[nI,3]), " " ,aCampos[nI,3])
                cPict   := Iif(Empty(aCampos[nI,4]), Nil ,aCampos[nI,4])
                cValid  := Iif(Empty(aCampos[nI,5]), ".t." ,aCampos[nI,5])
                cF3     := Iif(Empty(aCampos[nI,6]), NIL ,aCampos[nI,6])
                cWhen   := Iif(Empty(aCampos[nI,7]), ".T." ,aCampos[nI,7])

                nWidthG      := Iif(Empty(aCampos[nI,8]), 100,;
                    IiF(aCampos[nI,8]*3.5 > 100,100,nil))

```

```

cBlkSay := "||| OemToAnsi('"+cCaption+"') }"
cBlkGet := "{ | u | If( PCount() == 0,;
    "+cCampo+", "+cCampo+":= u ) }"
cBlkVld := "{ || "+cValid+"}"
cBlkWhen := "{ || "+cWhen+"}"

AADD(aSays,Array(1))
aSays[nI] := TSay():New(nLeft+1, nTop, &(cBlkSay), oScroll,,,,
    .F., .F., .F., .T.,,, 50, 8, .F., .F., .F., .F.,)
AADD(aGets,Array(1))

aGets[nI] := TGet():New( nLeft, nTop+50, &(cBlkGet), oScroll,;
    nWidthG, 11, cPict, &(cBlkVld),,,,.F.,, .T.,,,;
    .F., &(cBlkWhen), .F., .F.,, .F., .F., cF3,;
    (cCampo))

EndIf
Next
Endif

Return

```



### Anotações

---



---



---



---



---

## **Exemplo: Função TVisual ( MaWndBrowse com TEnchoice)**

```
/*
+-----+
| Função      | TVisual           | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição  | Enchoice para arquivos temporarios          |
+-----+
| Uso        | Curso ADVPL          |
+-----+
*/
USER FUNCTION TVisual(cAlias,nReg,nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk         := {||oDlg:End()}
LOCAL bCancel    := {||oDlg:End()}
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>"      ,{nLinha,nColuna}
//,"<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>,<Tamanho>})
// aCampos, {"TRB_FILIAL"           , "@!" , "Filial"      ,02}

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 ,nLinha := nLinha +015,nLinha) // Impar
        nColuna := IIF(nX%2 == 0,nColuna := nColuna+170,10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1],{nLinha,nColuna} ,
                    aCampos[nX][3],aCampos[nX][2],"AllwaysTrue()",;
                    "", ".F.",aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    & ("_" +aCampos[nX][1]) := (cAlias)->&(aCampos[nX][1])

Next nX

oDlg := TDialog():New(000,000,400,650,cCadastro,,,.T.)

TEnchoice(oDlg, aCposEnch,,,.T.)

oDlg:bInit := {|| EnchoiceBar(oDlg, bOk, bCancel,.F.,{},nReg,cAlias)}
oDlg:lCentered := .T.
oDlg:Activate()

RETURN
```

## **Exemplo: Função TInclui ( MaWndBrowse com TEnchoice)**

```
/*
+-----+
| Função      | TInclui           | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição   | Enchoice para arquivos temporarios          |
+-----+
| Uso         | Curso ADVPL          |
+-----+
*/

USER FUNCTION TInclui(cAlias,nReg,nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk
LOCAL bCancel    := {||oDlg:End()}
LOCAL aArea       := GetArea()
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>"      ,{nLinha,nColuna}
// ,,"<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>",<Tamanho>})
// aCampos,{ "TRB_FILIAL"           , "@!" , "Filial"      ,02}

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 ,nLinha := nLinha +015,nLinha) // Impar
        nColuna := IIF(nX%2 == 0,nColuna := nColuna+170,10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1],{nLinha,nColuna} ,
                      aCampos[nX][3],aCampos[nX][2],"AlwaysTrue()",;
                      "", ".T.",aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    Do Case
        Case aCampos[nX][5] == "C"
            & ("_" +aCampos[nX][1]) := Space(aCampos[nX][4])
        Case aCampos[nX][5] == "N"
            & ("_" +aCampos[nX][1]) := 0
        Case aCampos[nX][5] == "D"
            & ("_" +aCampos[nX][1]) := CTOD("")
        Case aCampos[nX][5] == "L"
            & ("_" +aCampos[nX][1]) := .F.
        Case aCampos[nX][5] == "M"
            & ("_" +aCampos[nX][1]) := Space(aCampos[nX][4])
    EndCase

    Next nX

    oDlg := TDialog():New(000,000,400,650,cCadastro,,,.T.)
```

```

TEnchoice(oDlg, aCposEnch, . . . . . T.)

bOk := { || IIF( U_TValid(cAlias, nReg, nOpcE, aCampos), ;
                  (U_TGravar(cAlias, nReg, nOpcE, aCampos), oDlg:End()), ) }

oDlg:bInit := { || EnchoiceBar(oDlg, bOk, bCancel, .F., {}, nReg, cAlias) }
oDlg:lCentered := .T.
oDlg:Activate()

RETURN

```

### **Exemplo: Função TAltera ( MaWndBrowse com TEnchoice)**

```

/*
+-----+
| Função | TAltera          | Autor | Arnaldo R. Junior | Data | |
+-----+
| Descrição | Enchoice para arquivos temporarios | |
+-----+
| Uso       | Curso ADVPL | |
+-----+
*/

USER FUNCTION TAltera(cAlias, nReg, nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk
LOCAL bCancel    := { || oDlg:End() }
LOCAL aArea       := GetArea()
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>" ,{nLinha,nColuna}
// ,"<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>",<Tamanho>})
// aCampos, {"TRB_FILIAL" , "@!" , "Filial" , 02}

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 , nLinha := nLinha +015, nLinha) // Impar
        nColuna := IIF(nX%2 == 0, nColuna := nColuna+170, 10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1], {nLinha,nColuna} ,
                      aCampos[nX][3], aCampos[nX][2], "AlwaysTrue()", ;
                      "", aCampos[nX][6], aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    & ("_" +aCampos[nX][1]) := (cAlias)->&(aCampos[nX][1])

Next nX

oDlg := TDIALOG():New(000,000,400,650,cCadastro, . . . . . T.)

```

```

TEnchoice(oDlg, aCposEnch, . . . . . T.)

bOk := { || IIF( U_TValid(cAlias, nReg, nOpcE, aCampos), ;
                  (U_TGravar(cAlias, nReg, nOpcE, aCampos), oDlg:End()), ) }

oDlg:bInit := { || EnchoiceBar(oDlg, bOk, bCancel, .F., {}, nReg, cAlias) }
oDlg:lCentered := .T.
oDlg:Activate()

RETURN

```

### **Exemplo: Função TExclui ( MaWndBrowse com TEnchoice)**

```

/*
+-----+
| Função | TExclui           | Autor | Arnaldo R. Junior | Data | |
+-----+
| Descrição | Enchoice para arquivos temporarios | |
+-----+
| Uso       | Curso ADVPL | |
+-----+
*/

USER FUNCTION TExclui(cAlias, nReg, nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk
LOCAL bCancel    := { || oDlg:End() }
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>" , {nLinha,nColuna}
// , "<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>",<Tamanho>})
// aCampos, {"TRB_FILIAL" , "@!" , "Filial" , 02}

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 , nLinha := nLinha +015, nLinha) // Impar
        nColuna := IIF(nX%2 == 0, nColuna := nColuna+170, 10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1], {nLinha,nColuna} ;
                  aCampos[nX][3], aCampos[nX][2], "AlwaysTrue()", ;
                  "", ".F.", aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    & ("_" +aCampos[nX][1]) := (cAlias)->&(aCampos[nX][1])

Next nX

oDlg := TDIALOG():New(000,000,400,650,cCadastro, . . . . . T.)

```

```

TEnchoice(oDlg, aCposEnch, . . . , T.)

bOk := { || IIF( U_TValid(cAlias, nReg, nOpcE, aCampos), ;
                  (U_TGravar(cAlias, nReg, nOpcE, aCampos), oDlg:End()), ) }

oDlg:bInit := { || EnchoiceBar(oDlg, bOk, bCancel, .F., {}, nReg, cAlias) }
oDlg:lCentered := .T.
oDlg:Activate()

RETURN

```

### **Exemplo: Função TValid ( MaWndBrowse com TEnchoice)**

```

/*
+-----+
| Função | TValid           | Autor | Arnaldo R. Junior | Data | |
+-----+
| Descrição | Enchoice para arquivos temporarios | |
+-----+
| Uso       | Curso ADVPL | |
+-----+
*/

USER FUNCTION TValid(cAlias, nReg, nOpc, aCampos)
LOCAL lRet
LOCAL nX
LOCAL nPosObrig := Len(aCampos[1])

For nX := 1 to Len(aCampos)
    IF aCampos[nX, nPosObrig] == .T.
        IF !(lRet := !Empty(&("_"+aCampos[nX, 1])))
            Help("TEnchoice", 1, "HELP", "OBRIGATORIO", "Existem campos
obrigatorios nao preenchidos", 1, 0)
            RETURN lRet // EXIT
        ENDIF
    ENDIF
Next nX

IF nOpc == 3
    IF !(lRet := !((cAlias)->(dbSeek(_TRB_FILIAL+_TRB_ID))))
        Help("TEnchoice", 1, "HELP", "INCLUSAO", "Ja existe um registro com esta
chave", 1, 0)
    ENDIF
ELSE
    IF !(lRet := (cAlias)->(dbSeek(_TRB_FILIAL+_TRB_ID)))
        Help("TEnchoice", 1, "HELP", "ALTERACAO", "Nao existe um registro com
esta chave", 1, 0)
    ENDIF
ENDIF

RETURN lRet

```

## **Exemplo: Função TGravar ( MaWndBrowse com TEnchoice)**

```
/*
+-----+
| Função | TGravar | Autor | Arnaldo R. Junior | Data | |
+-----+
| Descrição | Enchoice para arquivos temporarios | |
+-----+
| Uso | Curso ADVPL | |
+-----+
*/

USER FUNCTION TGravar(cAlias,nReg,nOpc,aCampos)
LOCAL nX

RecLock(cAlias,nOpc==3)
    IF nOpc == 5
        DbDelete()
    ELSE
        For nX := 1 to Len(aCampos)
            (cAlias)->&(aCampos[nX][1]) := & ("_" + aCampos[nX][1])
        Next nX
    ENDIF
Msunlock()

RETURN
```



### **Anotações**

---

---

---

---

## 6.2. DbType()

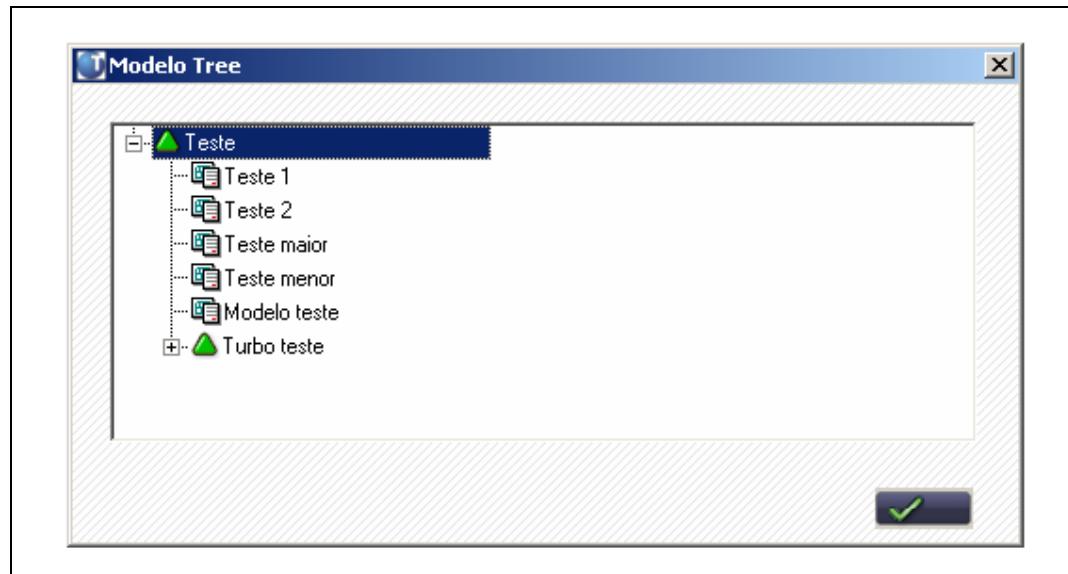
- Descrição:** Classe Advpl que permite a criação de um objeto visual do tipo Tree.
- Propriedades:**

<b>bChange</b>	Armazena o code-block executado quando da mudança de foco entre elementos do Tree atual.
----------------	--

- Métodos principais:**

<b>New</b>	Contrutor da Classe DbType. Retorna uma nova instância do Objeto da Classe DbType.
<b>AddTree</b>	Através do método AddTree, é possível acrescentar um 'nó' na árvore atual.
<b>AddTreeItem</b>	Uma vez acrescentado o nó, as próximas inclusões de itens na parvore serão realizadas neste nó, até que o mesmo seja fechado através do método EndTree.
<b>EndTree</b>	Através dele, podemos acrescentar itens na árvore atual ou último nó acrescentado em aberto.

- Aparência:**



## Método: New()

---

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

**Parâmetros:**

<b>nTop</b>	Coordenada vertical superior do Objeto.
<b>nLeft</b>	Coordenada horizontal esquerda do Objeto.
<b>nBottom</b>	Coordenada vertical inferior do Objeto.
<b>nRight</b>	Coordenada horizontal direita do Objeto.
<b>oWnd</b>	Janela pai do Objeto Tree
<b>bchange</b>	Code-Block contendo a ação a ser executada na mudança de foco entre os elementos da árvore.
<b>bRClick</b>	Code-Block a ser executado quando pressionado o botão direito do Mouse sobre um elemento da árvore.
<b>ICargo</b>	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.
<b>IDisable</b>	Se .T., cria o objeto do Tree desabilitado, não permitindo foco e navegação no mesmo quando ele não seja habilitado.

**Retorno:**

<b>Objeto</b>	Retorna uma nova instância do Objeto da Classe DbTree.
---------------	--

## Método: AddTree()

---

- Sintaxe:** oObj:AddTree ( < cLabel > , [ IPar02 ] , [ cResOpen ] , [ cResClose ] , [ cBMPOpen ] , [ cBMPClose ] , [ cCargo ] )

**Parâmetros:**

<b>cLabel</b>	Título do "nó" da árvore a ser acrescentado.
<b>IPar02</b>	(reservado)
<b>cResOpen</b>	Resource do RPO a ser utilizado quando o nó estiver aberto.
<b>cResClose</b>	Resource do RPO (bitmap) a ser utilizado quando o nó estiver fechado.
<b>cBMPOpen</b>	Path + Imagem bitmap a ser utilizada quando o nó estiver aberto.  A especificação de imagens a partir dos parametros cBMPOpen e cbMPClose apenas será considerada caso os parâmetros cResOpen e cResClose não forem especificados.
<b>cBMPClose</b>	Path + Imagem bitmap a ser utilizada quando o nó estiver fechado.  A especificação de imagens a partir dos parametros cBMPOpen e cbMPClose apenas será considerada caso os

	parâmetros cResOpen e cResClose não forem especificados.
<b>cCargo</b>	String com valor único, fornecido para identificar o nó atual. Pode ser recuperado posteriormente durante a navegação.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

#### **Método: AddTreeItem()**

---

- Sintaxe:** oObj:AddTreeItem ( <cLabel> , [cResource] , [cBitMap] , [cCargo] )

- Parâmetros:**

<b>cLabel</b>	Título do item da árvore.
<b>cResource</b>	Resource (BMP) do repositório, a ser mostrado à esquerda do título do item.
<b>cBitMap</b>	Arquivo de imagem (bitmap) a ser mostrado à esquerda do título do item. Este parâmetro apenas será considerado caso o parâmetro cResource não for informado.
<b>cCargo</b>	através dele, podemos especificar uma string de identificação para este item na árvore.

- Retorno:**

<b>Nenhum</b>	-
---------------	---

#### **Método: EndTree()**

---

- Sintaxe:** oObj:EndTree ( )

- Parâmetros:**

<b>Nenhum</b>	
---------------	--

- Retorno:**

<b>Nenhum</b>	
---------------	--

## **Exemplo:**

```
#include "Protheus.ch"

/*
+-----+
| Função | TDBTree       | Autor | MICROSIGA      | Data | |
+-----+
| Descrição | Exemplo de utilização da função DbTree | |
+-----+
| Uso       | Curso ADVPL | |
+-----+
/*

User Function TDBTree()
Local cBmp1 := "PMSEDT3"
Local cBmp2 := "PMSDOC"
Private cCadastro := "Meu Computador"
Private oDlg
Private oDBTree

DEFINE MSDIALOG oDlg TITLE cCadastro FROM 0,0 TO 240,500 PIXEL

oDBTree := dbTree():New(10,10,95,240,oDlg,{|| U_Proc(oDBTree:GetCargo()),;
, .T.}

    oDBTree:AddTree("Pentium 4"+Space(24),.T.,cBmp1,cBmp1,,,"1.0")

        oDBTree:AddTreeItem("Gabinete",cBmp2,,,"1.1")
        oDBTree:AddTreeItem("Monitor",cBmp2,,,"1.2")
        oDBTree:AddTreeItem("Teclado",cBmp2,,,"1.3")
        oDBTree:AddTreeItem("Mouse",cBmp2,,,"1.4")
        oDBTree:AddTreeItem("Som",cBmp2,,,"1.5")
        oDBTree:AddTree("Placa Mãe",.T.,cBmp1,cBmp1,,,"2.0")
            oDBTree:AddTreeItem("Processador",cBmp2,,,"2.1")
            oDBTree:AddTreeItem("Memória",cBmp2,,,"2.2")
            oDBTree:AddTreeItem("Vídeo",cBmp2,,,"2.3")
            oDBTree:AddTreeItem("Fonte",cBmp2,,,"2.4")

    oDBTree:EndTree()

oDBTree:EndTree()

DEFINE SBUTTON FROM 107,213 TYPE 1 ACTION oDlg:End() ENABLE OF oDlg
ACTIVATE MSDIALOG oDlg CENTER

Return
```

```

User Function Proc(cCargo)
Local cRet := ""

If      cCargo == "1.1"
MsgInfo("Gabinete Torre com 4 baias - Preto, com unidade de leitura e"+;
"gravação de CD/DVD",cCadastro)
Elseif cCargo == "1.2"
    MsgInfo("Monitor LCD 17' - LG",cCadastro)
Elseif cCargo == "1.3"
    MsgInfo("Teclado multimídia com funções de internet e e-mail",cCadastro)
Elseif cCargo == "1.4"
    MsgInfo("Mouse Optico sem fio",cCadastro)
Elseif cCargo == "1.5"
    MsgInfo("2 Caixas de Som - 50W RMS Cada",cCadastro)
Elseif cCargo == "2.1"
    MsgInfo("Processador Pentium 4 - 3.8 Ghz",cCadastro)
Elseif cCargo == "2.2"
    MsgInfo("1 Pente de Memória de 1Gb - DDR 500",cCadastro)
Elseif cCargo == "2.3"
    MsgInfo("Placa de Vídeo GeoForce 5000 com 256Mb",cCadastro)
Elseif cCargo == "2.4"
    MsgInfo("Fonte de Alimentação de 500W",cCadastro)
Endif

Return

```



### Anotações

---



---



---



---

### 6.3. MsSelect()

- Descrição:** A classe MsSelect cria um objeto browse (ou grid), com a primeira coluna sendo do tipo marcação.

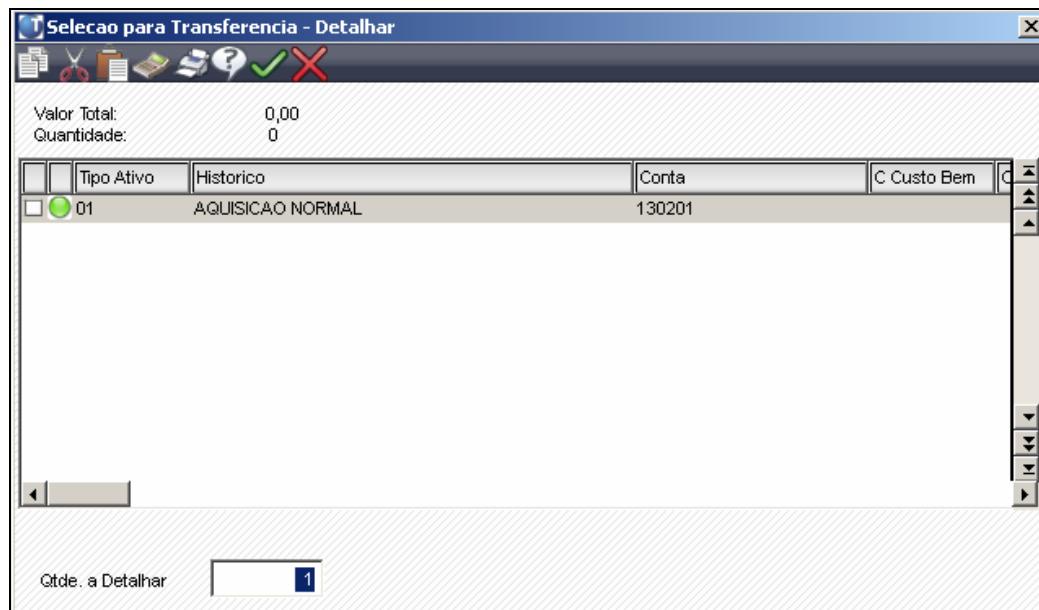
**Propriedades:**

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

**Métodos principais:**

New	Contrutor da Classe MsSelect. Retorna uma nova instância do Objeto da Classe MsSelect.
-----	---

**Aparência:**



*Anotações*

---

---

---

---

## Método: New()

---

- Sintaxe:** MsSelect():NEW ( < cAlias > , [ cCampo ] , [ cCpo ] , [ aCampos ] , [ lInv ] , [ cMar ] , [ aCord ] , [ cTopFun ] , [ cBotFun ] , [ oWnd ] , [ reserved ] , [ aColors ] )
- Parâmetros:**

<b>cAlias</b>	Tabela que será utilizada pela MsSelect
<b>cCampo</b>	Campo que será utilizado na coluna de marcação
<b>cCpo</b>	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
<b>aCampos</b>	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)
<b>lInv</b>	Indica se MsSelect irá utilizar marcação invertida
<b>cMar</b>	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.
<b>aCord</b>	Coordenadas para criação da MsSelect. Formato: 1 - Linha inicial 2 - Coluna inicial 3 - Linha final 4 - Coluna final
<b>cTopFun</b>	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.
<b>cBotFun</b>	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
<b>oWnd</b>	Objeto pai da MsSelect (MSDIALOG, MSWINDOW, MSPANEL, etc.)
<b>reserved</b>	Parametro reservado
<b>aColors</b>	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_MARRON BR_PINK BR_PRETO BR_VERDE BR_VERMELHO
--	---

**Retorno:**

<b>Objeto</b>	Retorna uma nova instância do Objeto da Classe MsSelect.
---------------	--

**Exemplo:**

```

DEFINE MSDIALOG oDlg1 TITLE "Selecao para Transferencia" From 9,0 To 32,80;
OF oMainWnd
@1.4,.8 Say "Valor Total:"
@1.4, 7 Say oValor      VAR nValor Picture "@E 999,999,999,999.99"
@1.9,.8 Say "Quantidade:"
@1.9, 9 Say oQtde      VAR nQtdBem Picture "@E 99999" SIZE 50,10
@1.4,15 Say Iif(MVParBox01==1,;
"Somente grupos de ["+MVParBox02+"] ate ["+MVParBox03+"]", "")
If cTipoDet == "D"
@12.4,01 Say "Qtde. a Detalhar"
@158,60 MSGET nQtde Picture "@E 999" SIZE 036, 10 OF oDlg PIXEL;
VALID nQtde > 0
ElseIf cTipoDet == "P"
@12.4,01 Say oLabel Var cLabel := "Tipo de Projeto"
oRad := TRadMenu():New(169,005, {"Industrial", "Florestal"},;
bSetGet(nOpcRad), oDlg,,{|| AF250RAD(nOpcRad)} ,,,100,12,,,.T.)
@15.5,01 Say oLabel2 Var cLabel2 := "Detalhamento:"
oRad2 := TRadMenu():New(210,005, {"Manter", "Alterar"},;
bSetGet(nOpcRad2), oDlg,,{|| AF250RAD2(nOpcRad2)} ,,,,
100,12,,,.T.)
@16.4,01 Say oLabel3 Var cLabel3 := "Percentual"
oGet := TGet():New(210,043,bSetGet(nPerc),oDlg,030,010,;
"@E 999.99",,,,.T.)
@18.2,01 Say oLabel4 Var cLabel4 := "Qtde. Det."
oGet2 := TGet():New(235,040,bSetGet(nQtde),oDlg,030,010,;
"@E 999",,,,.T.)
oGet:Hide()
oGet2:Hide()
oLabel3:Hide()
oLabel4:Hide()
EndIf
oMark := MsSelect():New("SN3", "N3_OK", "!N3_BAIXA",, @lInverte,;
@cMarca, {35,1,143,315})
oMark:bMark := {|| a250Display(cMarca, lInverte, oValor, oQtde) }

ACTIVATE MSDIALOG oDlg1
ON INIT EnchoiceBar(oDlg1,,;
{|| nOpct:=1,iif(DeParaEnt(),oDlg1:End(),.f.)}, {|| nOpct:=2,oDlg1:End() })

```

## 7. Introdução à relatórios gráficos

### 7.1. TReport()

#### 7.1.1. Introdução

##### Finalidade

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

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

Relatório de Compras										Folha: 1	Dt Ref: 09/10/98	Emissao: 09/10/98
<b>Vendedor:</b> Nome: 000002 KAU TOCUMA												
Período:	Inc. Total	Pedidos	Cliente	Nome	Dt-Consulta	Vencido-Digito	Dt-Balanço	Saida-Pagto	Pedido	M/ Total	V/ Total	%
	R\$1300	000002	RODRIGAM		09/10/98	00/10/98	/ /	10/10/98	1300,00	500,00	1	
	R\$1500	000001	TECCOM		09/10/98	10/10/98	09/10/98	10/10/98	1500,00	300,00	2	
	R\$1200	000002	TECCOM		09/10/98	10/10/98	09/10/98	10/10/98	1200,00	300,00	2	
	R\$1000	000002	RODRIGAM		09/10/98	00/10/98	/ /	10/10/98	1000,00	400,00	3	
<b>Total do Pedido por Vendedor:</b>												
000002	KAU TOCUMA									53.000,00	3.000,00	
<b>Total:</b>	<b>53.000,00</b>	<b>3.000,00</b>										
<b>Vendedor:</b> Nome: 000003 CELSO ALVAREZ												
Período:	Inc. Total	Pedidos	Cliente	Nome	Dt-Consulta	Vencido-Digito	Dt-Balanço	Saida-Pagto	Pedido	M/ Total	V/ Total	%
	R\$1000	000002	TECCOM		09/10/98	10/10/98	09/10/98	10/10/98	1000,00	200,00	2	
	R\$1000	000002	TECCOM		09/10/98	10/10/98	09/10/98	10/10/98	1000,00	200,00	2	
	R\$1000	000002	RODRIGAM		09/10/98	00/10/98	/ /	10/10/98	1000,00	1000,00	1	
	R\$1000	000002	RODRIGAM		09/10/98	00/10/98	/ /	10/10/98	1000,00	1000,00	1	
<b>Total do Pedido por Vendedor:</b>												
000003	CELSO ALVAREZ									1.700,00	400,00	
<b>Total:</b>	<b>1.700,00</b>	<b>400,00</b>										
<b>Total Geral:</b>												
										54.700,00	3.400,00	

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

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

## Descrição

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

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

Estrutura do componente TReport:

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

## Pré-Requisitos

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

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

```
#include "protheus.ch"

User Function MyReport()
Local oReport

If TRepInUse()      //verifica se a opção relatórios personalizáveis está
disponível
    Pergunte("MTR025", .F.)

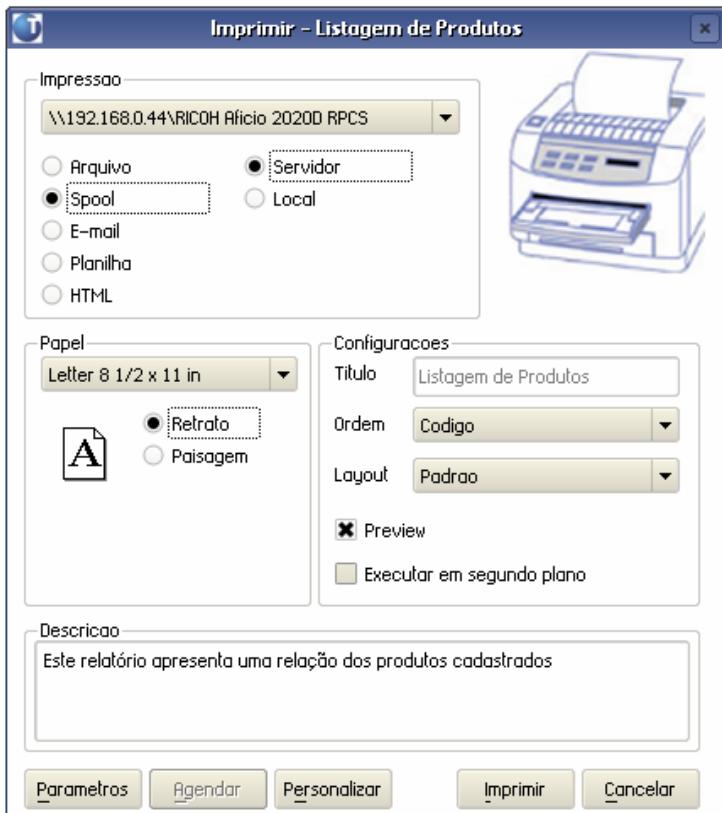
    oReport := ReportDef()
    oReport:PrintDialog()
EndIf
Return
```

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

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

## 7.1.2. Impressão do relatório personalizável

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



### 7.1.2.1. Parâmetros de impressão

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

#### Impressão

#### Arquivo

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

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

## **Spool**

---

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

## **E-mail**

---

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

**MV\_RELACNT**

Define a conta de e-mail para identificar a proveniência dos relatórios.  
Exemplo: relprotheus@microsiga.com.br

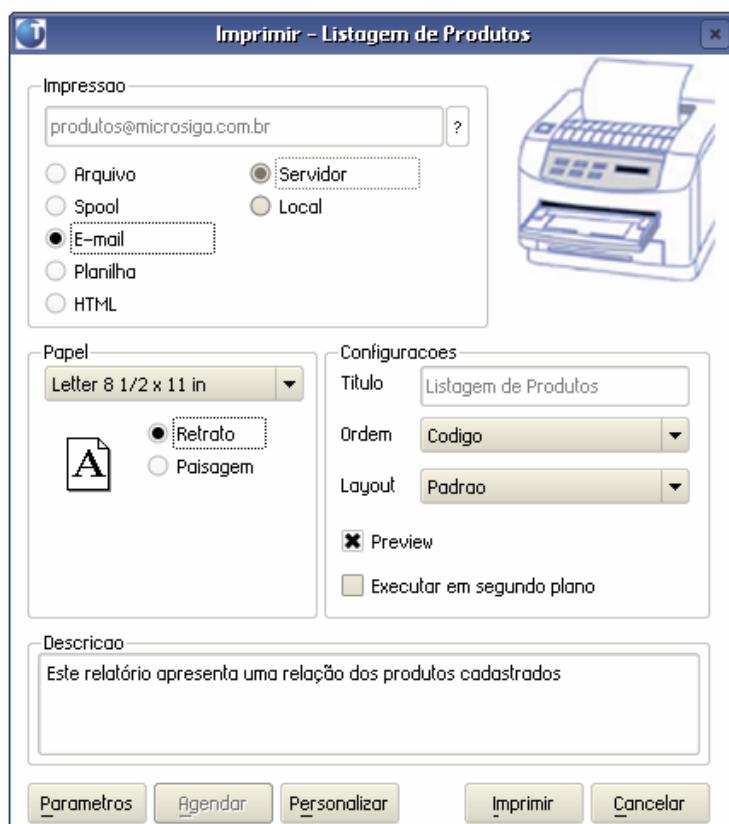
**MV\_RELPSW**

Define a senha da conta de e-mail para envio dos relatórios.

**MV\_RELSERV**

Define o servidor da conta de e-mail para o envio do relatório.  
Exemplo: smtp.microsiga.com.br

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



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

#### **Programação TelNet**

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

### **Papel**

#### **Tamanho do papel**

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

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

#### **Formato da impressão**

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

### **Configurações**

#### **Título**

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

#### **Ordem**

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

#### **Layout**

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

#### **Preview**

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

## **Executar em segundo plano**

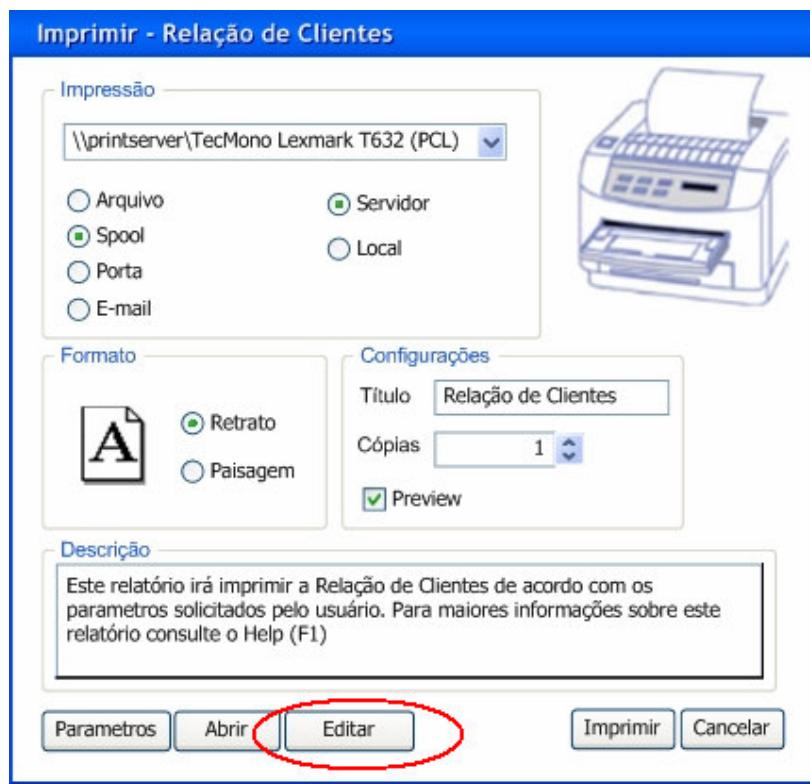
---

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

### **7.1.3. Personalização**

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

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



#### **7.1.3.1. Editando o layout do relatório**

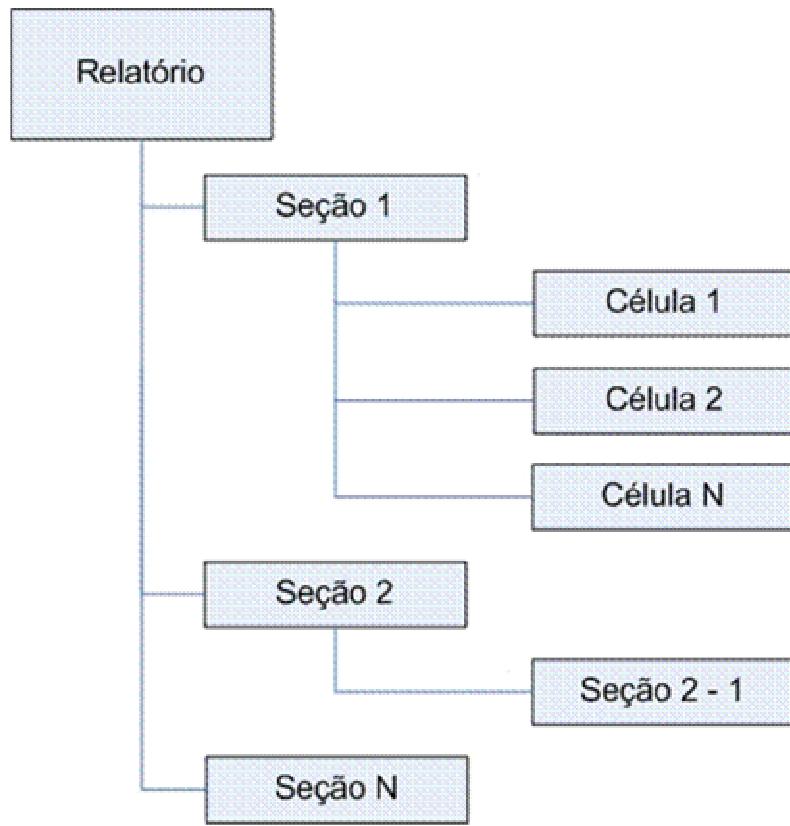
---

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

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

## **Nova estrutura do relatório TReport:**

---



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

### **7.1.4. Definindo a Função ReportDef()**

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

Os comandos que fará essa construção são:

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

## **DEFINE REPORT**

A função DEFINE REPORT é responsável pela criação do objeto Report, ou seja, o relatório.

Internamente, o DEFINE REPORT irá executar o método TReport():New().

Estrutura do componente TReport:

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

## **DEFINE SECTION**

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

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

Há a seção principal e as específicas.

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

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

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

## **DEFINE CELL**

Para cada seção, devem ser definidas as células. Célula é cada informação que deverá ser impressa. Pode ser um campo do cadastro, ou um resultado de uma operação. É uma Célula de impressão de uma seção (TRSection) de um relatório que utiliza a classe TReport

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

## MÓDULO 08: Aplicações ADVPL para o ERP

### 8. Protheus e o TOPCONNECT / TOTVS DbAccess

O TOTVSDBAccess é uma ferramenta de conectividade a bases de dados, permitindo que aplicativos acessem uma ampla variedade de servidores de banco de dados sem a necessidade de geração de códigos específicos para cada uma delas.

Os bancos de dados hoje suportados pelo TOTVSDBAccess são:

- Microsoft SQL7® ou superior
- Oracle 8® ou superior
- IBM Universal Database® (DB2)
- Informix®
- Sybase Adaptive Server®
- Postgres 7.1.1® ou superior
- MySQL®

O TOTVSDBAccess permite gerenciar as informações pertinentes à utilização do banco de dados, registrando e apresentando dados, como:

- Dados estatísticos de forma gráfica com relação à:
- Quantidade de usuários conectados
- Quantidade de tabelas em utilização
- IOs por segundo
- Máximo de IOs por segundo
- Relação dos usuários conectados
- Checagem das querys geradas pelos usuários conectados
- Quebra de conexão do usuário
- Mensagens de utilização do TOTVSDBAccess
- Configurações Gerais quanto a:
  - Banco de dados em utilização
  - Relação de usuários e senhas por banco de dados
  - Logins de usuário
  - Configuração de "Table Spaces" para determinados bancos de dados
  - Definição dos parâmetros de criação de tabelas para determinados bancos de dados
  - Validação de conexão, checando os parâmetros de conexão com o banco de dados
  - Monitoramento de índices gerados

## **8.1. Características técnicas com o uso do TOTVS DbAccess**

### **Comportamento de Queries – Colunas Calculadas**

Após enviar ao Banco de Dados uma solicitação de abertura de Query, em caso de sucesso na operação, o TOPConnect obtém do Banco de Dados a quantidade de colunas que serão retornadas, e qual é o tamanho e tipo e especificações de cada uma das colunas, para retornar ao Protheus uma estrutura definida de informações, e o cursor aberto é lido conforme são solicitados os próximos registros ( DBSkip() no Advpl ), e a cada requisição, uma linha do cursor é retornada, até não haver mais linhas no cursor, atingindo o final do arquivo no Advpl ( EOF() ).

Quando especificamos uma coluna de retorno na Query, que corresponde a informações lidas diretamente de uma ou mais coluna(s) da tabela, cada coluna retornada pela Query contém uma definição de tipo e tamanho correspondente ao campo de origem do dado de retorno. Deste modo, se considerarmos, por exemplo, a Query : SELECT A1\_FILIAL FROM SA1990 , onde o campo A1\_FILIAL na tabela SA1990 é um campo do tipo varchar, com 2 bytes de tamanho, quando o cursor for aberto no Banco de Dados, o TOPConnect irá obter do banco que o result-set gerado possui apenas uma coluna, e esta coluna é do tipo varchar, com no máximo 2 bytes de tamanho.

Os Bancos de Dados não apenas permitem recuperar retornos diretamente lidos de uma ou mais tabelas, mas também é possível abrir um cursor onde uma ou mais colunas contenham um resultado calculado, utilizando-se de expressões e/ou funções nativas do Banco de Dados. Para estes tipos de retorno, cada banco de dados pode apresentar comportamentos diferenciados ao descrever o tamanho de retorno da(s) coluna(s).

Este comportamento pode variar de acordo com o Banco de Dados, a expressão e/ou funções utilizadas para gerar a coluna de retorno, e inclusive podem ter diferenças entre versões e Build's do mesmo Banco de Dados e/ou parametrizações específicas do Banco de Dados e/ou da aplicação Client / ODBC / API utilizada para acesso ao Banco.

Para ilustrar melhor estas diferenças, vejamos abaixo os resultados obtidos com queries, em bancos distintos, onde foi criada uma tabela com a mesma estrutura, e 10 linhas de dados, e nela foram executadas algumas querys, onde temos como retorno uma coluna calculada. A tabela possui os campos CPOC, tipo Caractere, tamanho 10 bytes, numerados de 0000000001 a 0000000010 e um campo numérico CPON, alimentado com valores positivos. Para cada Query, mostramos o valor retornado ao Advpl, contendo o tipo de dado retornado e o tamanho (len) da coluna, no formato . ( Tipo ( tamanho ) ).

#### **Query: SELECT MIN(CPOC) CPOTESTE FROM TESTRET**

- Retorno:** Todos os bancos retornaram uma coluna com tamanho de 10 bytes.
- Bancos:**

<b>Database</b>	<b>Tipo</b>	<b>Tamanho</b>	<b>Conteúdo</b>
<b>MSSQL 2000</b>	Caracter	10	0000000001
<b>INFORMIX</b>	Caracter	10	0000000001
<b>ORACLE</b>	Caracter	10	0000000001
<b>MYSQL</b>	Caracter	10	0000000001
<b>DB2</b>	Caracter	10	0000000001

**Query:** `SELECT MIN(CPOC) CPOTESTE FROM TESTRET WHERE CPON < 0`

---

- Retorno:** Todos os bancos retornaram uma coluna vazia.
- Bancos:**

Database	Tipo	Tamanho	Conteúdo
<b>MSSQL 2000</b>	Caracter	10	Vazio
<b>POSTGRES</b>	<b>Caracter</b>	<b>0</b>	<b>Vazio</b>
<b>INFORMIX</b>	Caracter	10	Vazio
<b>ORACLE</b>	Caracter	10	Vazio
<b>MYSQL</b>	Caracter	10	Vazio
<b>DB2</b>	Caracter	10	Vazio

**Query:** `SELECT LEFT(CPOC,5) CPOTESTE FROM TESTRET`

---

- Retorno:** Todos os bancos que suportam a função LEFT() retornaram uma coluna de informações, onde os 5 primeiros bytes eram '00000'.
- Observações:** Os bancos retornaram que a coluna tem 5 bytes de tamanho. Porém, o Banco DB2 informa que a coluna têm tamanho de 4000 Bytes. Este mesmo retorno foi obtido no DB2 quando utilizadas funções RIGHT(), REPLACE() e REPEAT(), no banco DB2, e para o Banco Postgres, foi retornado o tamanho de 2000 bytes para estas funções.

- Bancos:**

Database	Tipo	Tamanho	Conteúdo
<b>MSSQL 2000</b>	Caracter	5	00000
<b>MYSQL</b>	Caracter	5	00000
<b>DB2</b>	<b>Caracter</b>	<b>4000</b>	<b>00000 ...</b>

**Query:** `SELECT SUBSTRING(CPOC,2,4) CPOTESTE FROM TESTRET`

---

- Retorno:** Todos os bancos retornaram uma coluna de tamanho 4 bytes, exceto o POSTGRES, que retornou tamanho de 2000 bytes.
- Bancos:**

Database	Tipo	Tamanho	Conteúdo
<b>MSSQL 2000</b>	Caracter	4	0000
<b>POSTGRES</b>	Caracter	2000	0000 ...
<b>INFORMIX</b>	Caracter	4	0000
<b>ORACLE</b>	Caracter	4	0000
<b>MYSQL</b>	Caracter	4	0000
<b>DB2</b>	Caracter	4	0000

Devido a estas diferenças de comportamento, devemos tomar alguns cuidados ao desenvolver aplicações que utilizam colunas calculadas em queries. Nestes casos, deve ser utilizada uma função do banco de dados, para assegurar que o banco de dados retorne o tamanho de coluna adequado ao TOPConnect / TotvsDBAccess.

Por exemplo, vejamos a query abaixo :

```
SELECT REPLACE(CPOC, '0', '_') AS CPOTESTE FROM TESTRET
```

Se já é de conhecimento do programador que o campo da base de dados CPOC desta tabela tem 10 bytes de tamanho, devemos fazer um cast deste retorno, para char ou varchar, com tamanho de 10 bytes. Dessa forma, o banco retorna o tamanho esperado da coluna para o TOPConnect quando da abertura da query. Isto evita que espaços desnecessários trafeguem pela rede para atender à requisição, e que este valor excedente seja usado para alimentar uma variável do Advpl. Vejamos como usar este recurso na query acima descrita, utilizando uma sintaxe aceita pela maioria dos Bancos de Dados homologados :

```
SELECT CAST( REPLACE(CPOC, '0', '_') AS VARCHAR(10)) CPOTESTE FROM TESTRET
```

Vale a pena salientar que, cada banco de dados possui, com pequenas variações, uma sintaxe para permitir o CAST de um retorno. Para maiores detalhes, a documentação do banco deve ser consultada.

### **Comportamento diferenciado com Bandos de Dados PostGres**

O Banco de Dados Postgres possuem um comportamento diferenciado entre os Builds do Server do Banco, e possui uma parametrização na configuração do ODBC que pode interferir na maneira como os atributos das colunas de retorno de uma query são retornados.

Em versões inferiores à Postgres 8.x, as diferenças de comportamento em relação às demais Bancos de Dados possuem um diferencial muito significativo, em praticamente todas as funções de retorno calculado. Para estes casos, em se tratando de Postgres versões inferiores a 8.x, a precisão de retorno da coluna corresponde ao tamanho máximo de retorno do tipo VarChar. ( Este tamanho máximo pode ser configurado na conexão ODBC do PostgreSQL, e o valor default é 65536 bytes. )

Na função Substring(), mesmo que esteja especificado no 3. parâmetro da função, a quantidade final limitada de Bytes, o Banco de Dados descreve esta coluna de retorno como sendo um VarChar() com a precisão do maior valor VARCHAR() a ser retornado pelo client. Na função Max(), quando especificado um campo da base do tipo Char ou Varchar, temos um comportamento semelhante.

Quando foram homologados os Bancos Postgres para uso com o ERP Microsiga e TOPConnect 4, praticamente não eram utilizadas queries com colunas calculadas nas aplicações Advpl, e este comportamento diferenciado não foi percebido. Conforme as rotinas foram evoluindo, buscando mais performance através do uso de Queries, tornou-se perceptível estas diferenças operacionais.

Algumas destas diferenças operacionais foram assimiladas pelas funções do ERP, por exemplo a ChangeQuery, que em versão atualizada (\*\* Protheus 8 e P10, fonte aplib070.prw com data igual ou superior a 08/03/2008) , aplicam o CAST() automaticamente para SELECT MAX(CAMPO), quando o banco é Postgres, e o campo selecionado for um campo da base de dados, presente no Dicionário de Campos ( SX3 ).

Para todos os efeitos, para utilização com as versões homologadas de Postgres inferiores a 8.x, é necessário alterar a configuração de ODBC do Postgres, para limitar o retorno do tipo VarChar para 2000 Bytes, pois o retorno do valor default ( 65536 bytes ) não é suportado pelo TOPConnect / TOTVSDBAccess. Alteramos esta configuração no arquivo .odbc.ini no linux, e/ou no arquivo de configuração do ODBC do Postgres utilizado, inserindo na seção da conexão / identificação do banco a chave abaixo :

```
MaxLongVarcharSize=2000
```

Atualmente está em processo de homologação as versões mais recentes (8.x) do banco Posgres. Quando homologado, a informação estará disponível na DEM, em Principal -> Guia de Referência -> Guia de Plataformas Homologadas -> Bancos de Dados Relacional X S.O.

### **Conceito de Índices Permanentes e Diferenças das RDDs**

Na RDD TOPCONN, não temos o conceito de criação de índice temporário, apenas índice permanente. Um índice permanente é criado fisicamente no Banco de Dados relacional utilizado, através do TOPConnect / DbAccess, onde devemos especificar um ou mais campos para compor a chave / expressão de índice.

Quando utilizamos CodeBase/DBF, podemos especificar como campos de chave de índice o retorno de funções, como por exemplo STR(), DTOS(), SUBSTR(), entre outras suportadas de modo nativo pela aplicação provedora de acesso, no caso ADS Local e/ou ADS Server.

Quando usamos um Banco Relacional SGDB através do TOPConnect / DbAccess, devemos especificar uma expressão de chave que sempre retorne um valor Caractere, e as únicas funções permitidas para adequações de tipo de dado no Advpl para estas expressões de índice são : DTOS() e STR().

### **Funcionamento Interno**

A expressão utilizada para a criação de um índice permanente em Advpl, por exemplo : CPOC + DTOS(CPOD) + STR( CPON,10 ), quando utilizamos o RDD TOPCONN, será ajustada para o Banco de Dados relacional utilizado para uma expressão contendo apenas a lista de campos desejada, na sequência em que foi especificada, e as funções DTOS() e STR() não são passadas ao banco de dados, pois não existe a necessidade de conversão de dados para o Banco.

Um campo do tipo 'D' Data é tratado pelo TOPConnect e gravado na tabela em questão como um campo 'C' Caractere, de 10 bytes, no formato AAAAMMDD, e os números são gravados em um campo DOUBLE\*.

## **Quebra de Compatibilidade com CodeBase/DBF**

---

Os Bancos relacionais, em sua grande maioria, senão todos, suportam apenas a criação de índices onde sejam especificados campos físicos da base de dados. Não são suportadas funções de conversão ou transformação de dados na criação de índices, como por exemplo substring(), left(), entre outras. Embora alguns bancos permitam a criação de colunas calculadas, as mesmas são 'virtuais', isto é, não existem fisicamente na tabela, e também não é permitido o uso de colunas virtuais para a criação de índices.

Entendemos que o RDD CodeBase / DBF, onde originalmente foi desenvolvido o ERP, nos dava certa flexibilidade ao permitir operações como estas, porém a um custo de processamento mais alto, pois para cada inserção ou alteração, o RDD tem que executar funções de conversao e concatenação de strings para atualização dos índices que usam este recurso.

Manter o suporte a todos os indices permanentes, criados a partir de resultados de expressões, nas versões anteriores do ERP, teríamos um custo muito alto de complexidade, performance e duplicidade de informações nas tabelas. Seria necessário criar colunas físicas nas tabelas, transparentes ao usuario, para manter copias em duplicidade de partes de colunas agrupadas, com gatilhos do banco de dados disparados internamente em operações de insert e update, um mecanismo relativamente complexo de se manter, instável para dar manutenção, e custoso em performance para o Banco de Dados.

Partindo da premissa que, se uma determinada informação deve ser indexada para busca, ela deve ser uma informação que ocupa exclusivamente um campo físico da base de dados, são evitadas as operações constantes de concatenação e desmembramento de uma informação agrupada, colocando cada parte da informação em seu devido espaço ( campo ) definido, mesmo com o impacto gerado para alterar as aplicações que usavam estes tipos de índices, os ganhos obtidos em organização e performance foram muito significativos.

## **Lista dos códigos de erro do TOPConnect / DbAccess**

Quanto utilizado o TOPConnect / DbAccess para conexão e operações com Bancos de Dados, as ocorrências de erro são reportadas ao Protheus informando em conjunto com o erro um número menor que zero, correspondendo a um tipo de erro.

Para a maioria das ocorrências, relacionadas ao Banco de Dados, deve-se olhar o log de erro gravado pelo TOPConnect / DbAccess ( arquivo topconn.log ) para obter maiores detalhes sobre a ocorrência.

<b>TC_NO_ERROR</b>	0	<b>COMM_INITPGM_ERROR</b>	-81
<b>NO_ROUTER_INSTALLED</b>	-1	<b>COMM_PARAM_ERROR</b>	-86
<b>NO_CONNECTION</b>	-2	<b>COMM_PROGRAM_ERROR</b>	-88
<b>NO_USER_SECURITY</b>	-4	<b>COMM_INSMEM_ERROR</b>	-90
<b>PASSTHRU_FAILED</b>	-5	<b>INVALID_BUILD</b>	-99
<b>NO_MORE_CONNECTIONS</b>	-6	<b>INVALID_TOPAPI</b>	-100
<b>INVALID_TOP_KEY</b>	-8		
<b>INVALID_ENVIRONMENT</b>	-9		
<b>INVALID_FILE</b>	-10		
<b>UNKNOWN_FILE</b>	-11		
<b>EXCLUSIVE_REQUIRED</b>	-11		
<b>INVALID_OPERATION</b>	-13		
<b>INVALID_KEY_NUM</b>	-14		
<b>FILE_IN_USE</b>	-15		
<b>TOO_MANY_FILES</b>	-16		
<b>INVALID_NUMRECS</b>	-17		
<b>CALL_FAILED</b>	-18		
<b>COMMAND_FAILED</b>	-19		
<b>OVERRIDE_FAILED</b>	-20		
<b>QUERY_FAILED</b>	-21		
<b>CREATION_FAILED</b>	-22		
<b>OPEN_FAILED</b>	-23		
<b>NOT_OPENED</b>	-24		
<b>NO_RECORD_FOUND</b>	-25		
<b>END_OF_RECORDS</b>	-26		
<b>NO_WRITE_POSSIBLE</b>	-27		
<b>NO_RECORD_EQUAL</b>	-28		
<b>UPDATE_FAILED</b>	-29		
<b>DELETE_FAILED</b>	-30		
<b>RECORD_LOCKED</b>	-31		
<b>FILE_LOCKED</b>	-32		
<b>NO_AUTORIZATION</b>	-33		
<b>TOO_MANY_USERS</b>	-34		
<b>NO_DB_CONNECTION</b>	-35		
<b>NO_CONN_ALLOWED</b>	-36		
<b>INTEGRITY_FAILURE</b>	-37		
<b>BUFFER_OVERFLOW</b>	-40		
<b>INVALID_PARAMETERS</b>	-41		
<b>NO_AUDIT_CONNECTION</b>	-50		
<b>COMM_DOSMEM_ERROR</b>	-58		
<b>COMM_PARTNER_ERROR</b>	-67		
<b>COMM SNDSTAT_ERROR</b>	-76		
<b>COMM RCVSTAT_ERROR</b>	-76		

## **8.2. Funções ADVPL para TOPCONNECT / TOTVS DbAccess**

---

Neste tópico serão descritas as funções da linguagem ADVPL que permitem a interação com a aplicação TopConnect / DbAccess.

### **Listas das funções de interação com a aplicação TopConnect / DbAccess:**

---

- TCCANOPEN**
- TCCONTYPE**
- TCDELFILE**
- TCGENQRY**
- TCGETDB**
- TCLINK**
- TCQUERY**
- TCQUIT**
- TCSETCONN**
- TCSETFIELD**
- TCSPEXEC**
- TCSPEXIST**
- TCSQLERROR**
- TCSQLEXEC**
- TCSRVTYPE**
- TCUNLINK**
- TCCHKOBJ**
- TCEXEERROR**
- TCPGMEXE**
- TCSYSEXE**

### **Listas das funções acessórias utilizadas nos fontes como facilitadoras:**

---

- CHANGEQUERY**
- RETFULLNAME**
- RETSQLCOND**
- RETSQLNAME**
- RETSQTABLE**
- SQLCOPY**
- SQLORDER**
- SQLTOTRB**

## **Funções de interação com o TopConnect / DbAccess**

### **TCCANOPEN ()**

Verifica a existência de tabelas e índices no servidor.

- Sintaxe:** TCCanOpen(**cTable** , **cIndice**)

- Parâmetros:**

<b>cTable</b>	Nome completo da tabela no banco de dados.
<b>cIndice</b>	Opcional, indica o nome completo do índice referente a tabela especificada.

- Retorno:**

<b>Lógico</b>	Indica a existência da tabela e/ou do índice especificado.
---------------	--

- Exemplo:**

```
//Verifica a existência da tabela customer
IF !TCCanOpen("CUSTOMER")
    dbCreate("CUSTOMER", aStru, "TOPCONN")
ENDIF
USE CUSTOMER SHARED NEW VIA "TOPCONN"

//Verifica a existência do índice
IF !TCCanOpen("CUSTOMER", "CUSTCOD")
    INDEX ON CODIGO TO CUSTCOD
ELSE
    SET INDEX TO CUSTCOD
ENDIF
```

### **TCCONTYPE()**

Especifica o protocolo de comunicação a ser utilizado pelo TOPConnect.

- Sintaxe:** TCConType(**cType**)

- Parâmetros:**

<b>cType</b>	Tipo de protocolo de comunicação a ser utilizado pelo TopConnect, aonde:  TCPIP: Utiliza o protocolo TCP/IP. NPIPE: Utiliza o protocolo Named Pipes. APPC: Utiliza o protocolo APPC. BRIDGE: Utiliza o utilitário TOPBrigde para comunicação de estações DOS NPIPE com servidor AS/400 utilizando TCP/IP.
--------------	--

- Retorno:**

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

- Exemplo:**

```
#INCLUDE "TOPCONN.CH"
//Especifica conexão TCP/IP
TCConType("TCPIP")

//Conecta-se ao ambiente SIGAADV no Microsoft SQL-Server
TCLink("MSSQL/SIGAADV")
```

## **TCDELFILE()**

Exclui uma tabela no servidor.

- Sintaxe: TCDelFile(cTable)**

- Parâmetros:**

cTable	Nome completo da tabela a ser excluída.
--------	---

- Retorno:**

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

- Exemplo:**

```
TCDelFile("CUSTOMER")
```



*Anotações*

---

---

---

---

## **TCGENQRY()**

Permite a execução de uma query de seleção no banco de dados e retorna um recordset com o resultado.

- Sintaxe:** **TCGENQRY(xParam1,xParam2,cQuery)**

- Parâmetros:**

<b>xParam1</b>	Parâmetro reservado.
<b>xParam2</b>	Parâmetro reservado.
<b>cQuery</b>	Query de seleção com sintaxe no padrão SQL ANSI a ser executada no banco de dados.

- Retorno:**

<b>RecordSet</b>	RecordSet contendo o resultado da execução da query de seleção.
------------------	---



*Importante*

Para que os dados contidos no RecordSet retornado pela função TCGenQry() sejam utilizados pela aplicação ADVPL é necessário disponibilizá-los em uma WorkArea.

Desta forma a função TCGenQry() deverá ser utilizada em conjunto com a função DbUseArea(), para que a mesma disponibilize o RecordSet como uma área de trabalho do ADVPL.

- Exemplo:**

```
cQuery := "SELECT DISTINCT CV8_PROC FROM "+RETSQLNAME("CV8")
cQuery += " WHERE "
cQuery += "CV8_FILIAL = '" +MV_PAR01+ "' AND "
cQuery += "D_E_L_E_T_ = ' ' "
cQuery += "ORDER BY CV8_PROC"
cQuery := ChangeQuery(cQuery)
dbUseArea(.T., "TOPCONN", TCGenQry(, , cQuery), "CV8QRY", .F., .T.)
```

## **TCGETDB()**

Retorna o tipo de Banco de Dados corrente.

- Sintaxe:** **TCGETDB()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Caracter</b>	Descrição referente ao tipo do banco de dados corrente, aonde:  ANYWHERE - Sybase SQL Anywhere ANYSYBASE - Sybase SQL Anywhere (Emulando Sybase SQL-Server) DB2 - IBM Universal Database INFORMIX - Informix MSSQL - Microsoft SQL-Server 6.5 MSSQL7 - Microsoft SQL-Server 7.0 ou superior ORACLE - Oracle POSTGRES - PostGres SYBASE - Sybase SQL-Server  Nota: O retorno da função pode variar de acordo com a homologação de novos databases para utilização com a ferramenta TopConnect / DbAccess.
-----------------	--

**Exemplo:**

```
cDataBase := TCGETDB() // "MSSQL7"
```

---

### **TCLINK()**

Abre uma conexão com o Servidor TOPConnect.

**Sintaxe: TCLink(cConnectString,cServer)**

**Parâmetros:**

<b>cConnectString</b>	String composta conforme a regra: Para TOPConnect NT : Nome do Banco de Dados / Nome do Ambiente  Para TOPConnect/400: Nome do Ambiente
<b>cServer</b>	Nome do servidor conforme a regra: Para conexões NamedPipe, APPC e Brigde : O nome do servidor TOPConnect.  Para conexões TCP/IP : O endereço IP do servidor TOPConnect

**Observações:**

- Para o TOPConnect NT acessando Oracle, um ambiente é o nome do Alias criado com o utilitário SQL-Net.
- Para o TOPConnect NT acessando os demais SGDB's um ambiente é um System DSN criado através do ODBC Data Source Administrator.
- Para o TOPConnect AS/400 um ambiente é uma Library criada através do comando CRTTOPENV do TOPConnect.

- Para informações sobre a criação de ambientes consulte o manual de instalação do TOPConnect.
- A lista dos tipos de bancos de dados suportados pela aplicação TopConnect / DbAccess é a mesma das possibilidades de retorno da função TCGETDB conforme abaixo:

ANYWHERE	- Sybase SQL Anywhere
ANYSYBASE	- Sybase SQL Anywhere (Emulando Sybase SQL-Server)
DB2	- IBM Universal Database
INFORMIX	- Informix
MSSQL	- Microsoft SQL-Server 6.5
MSSQL7	- Microsoft SQL-Server 7.0 ou superior
ORACLE	- Oracle
POSTGRES	- PostGres
SYBASE	- Sybase SQL-Server

**Retorno:**

<b>Numérico</b>	A função TCLink retorna ou o número da conexão ou um valor negativo contendo o código do erro.  A relações dos códigos de erros ocorridos durante uma tentativa de conexão está disponível no tópico 8.1. deste material.
-----------------	---

**Exemplo:**

```
//Conecta-se ao Microsoft SQL-Server no ambiente SIGAADV
//Protocolo Named Pipes
TCConType("NPIPE")

nCon := TCLink("MSSQL/SIGAADV", "TOPSRV")
if nCon < 0 //Conexões com retorno < 0 significam erro
    Alert("Falha de conexão com o TOPConnect")
endif
//Protocolo TCP/IP
TCConType("TCPIP")
//Conecta-se ao Oracle - no ambiente TESTES
nCon := TCLink("ORACLE/TESTES", 172.16.1.2)

//Protocolo APPC
//Conecta-se ao AS/400 no ambiente PRODUCAO
nCon := TCLink("PRODUCAO", "TOP400")
```



*Anotações*

---



---



---



---



---

## **TCQUERY()**

Executa uma Query no servidor e coloca seu retorno em uma WorkArea.



*Importante*

Durante o processo de compilação, a sintaxe TCQUERY() é substituída pelas expressões:

dbUseArea(.T., "TOPCONN", TcGenQry(,,cQuery), "ALIAS" ,.T.,.F.)

Esta substituição é realizada conforme as definições do include TOPCONN.CH.

Desta forma é recomendável a utilização direta da sintaxe DbUseArea() + TcGeQry().

**Sintaxe: TCQUERY cSQLExpr ALIAS cAlias NEW VIA "TOPCONN"**

**Parâmetros:**

<b>cSQLExpr</b>	Expressão SQL a ser enviada ao servidor.
<b>ALIAS cAlias</b>	Especifica um nome para a Work Area a ser aberta.
<b>NEW</b>	Abre a tabela <cTable> na próxima Work Area disponível. Se esta clausula não for especificada, <cTable> será na Work Area corrente.
<b>VIA "TOPCONN"</b>	Este parâmetro indica ao ADVPL que esta Work Area será gerenciada pelo TOPconnect.

**Observações:**

- A Work Area criada com o comando TCQUERY é READ-ONLY, portanto não é permitido o uso de APPEND's ou REPLACE's.
- O TOPConnect tem apenas um cursor para a Query executada portando apenas os seguintes comandos são suportados em uma Work Area criada com TCQUERY:
  - GO TOP - DbGoTop()
  - GO BOTTOM - DbGoBottom()
  - SKIP - DbSkip()
  - Eof()
  - Bof()
- Ao executar uma Query que retorne um ou mais valores calculados, o nome dos campos da WorkArea serão COLUMN1, COLUMN2... COLUMNn.

**Retorno:**

**Nenhum**

.

**Exemplo:**

```
//Abre a tabela de Clientes em uma nova WorkArea  
  
cQuery := "SELECT a.codigo, b.nome FROM CLIENTES a, CLIDATA b "  
cQuery += "WHERE a.CODIGO = b.CODIGO ORDER BY CODIGO,NOME "  
  
TCQUERY cQuery ALIAS CLIENTES NEW VIA "TOPCONN"  
dbGoTop()  
  
While !Eof()  
    ALERT(CODIGO+" | "+NOME)  
    dbSkip()  
end  
  
cQuery := 'SELECT SUM(VALOR),SUM(SALDO) FROM CUSTOMER'  
TCQUERY cQuery NEW VIA "TOPCONN"  
ALERT(CValToChar(COLUMN1)+" | "+CValToChar(COLUMN2))  
// COLUMN1 - Somatoria dos valores  
// COLUMN2 - Somatoria dos saldos
```

---

## **TCQUIT()**

---

Encerra todas as conexões com o TOPConnect.

**Sintaxe: TCQuit()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Nenhum</b>	.
---------------	---

---

## **TCSETCONN()**

---

Seleciona a conexão ativa.

**Sintaxe: TCSETCONN(nConn)**

**Parâmetros:**

<b>nConn</b>	Número da conexão
--------------	-------------------

**Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
// Abre conexão com o ambiente de Produção
nCon1 := TCLink("MSSQL/PRODUCAO")
if nCon1 < 0
    Alert("Falha conectando ambiente de Produção")
    QUIT
endif

// Abre conexão com o ambiente de Testes
nCon2 := TCLink("MSSQL/TESTES")

if nCon2 < 0 then
    Alert("falha conectando ambiente de Testes")
    QUIT
endif

TCSetConn(nCon1)

//Abre tabela de Clientes no ambiente de produção
USE CUSTOMER ALIAS PROD SHARED NEW VIA "TOPCONN"

TCSetConn(nCon2)

//Abre tabela de Clientes no ambiente de testes
USE CUSTOMER ALIAS TEST SHARED NEW VIA "TOPCONN"
...
```

## **TCSETFIELD()**

Esta função serve como apoio ao comando TCQUERY, na recuperação de campos tipo NUMERIC, DATE e LOGICAL, pois os mesmos são gravados fisicamente no Banco de Dados como caracteres, e no caso dos numéricos como float.

**Sintaxe: TCSetField(cAlias, cField ,cType, nTam, nDec )**

**Parâmetros:**

<b>cAlias</b>	Alias da WorkArea gerada pela execução da query.
<b>cField</b>	Nome do campo a ser tratado
<b>cType</b>	Tipo desejado para o campo
<b>nTam</b>	Tamanho total desejado para o campo
<b>nDec</b>	Número de decimais desejado para o campo

**Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo 01:**

```
TCQUERY "SELECT NOME, DATA, MARRIED, VALOR FROM CUSTOMER" ALIAS QUERY VIA  
"TOPCONN"  
  
TCSetField("QUERY", "DATA", "D")  
TCSetField("QUERY", "VALOR", "N", 12, 2)
```



*Importante*

Dentro de um programa ADVPL já podemos considerar como um campo data não mais um campo caracter como é o seu armazenamento.

O campo Data só é retornado como um campo caracter por que estamos utilizando a função TCQUERY, se não o tratamento é automático.



*Dica*

A estrutura dos parâmetros da função TCSETFIELD() é a mesma retornada pela função DbStruct() e que é utilizada em diversas funções que manipulam arquivos de dados.

Desta forma é muito comum a montagem de queries em tabelas do sistema partindo da estrutura de campos retornada pela função DbStruct() sendo que após a execução dos campos Tcquery() ou DbUseArea() normalmente é utilizado o campo TcSetField com base no conteúdo deste mesmo array de estrutura.

**Exemplo 02:**

```
Local cQuery      := ""  
Local cCampos     := ""  
Local aStruSA1    := SA1->(DbStruct())  
Local nX          := 0  
Local nMax        := Len(aStruSA1)  
  
// Monta a String cCampos de acordo com o conteúdo de aStruSA1  
  
aEval(aStruSA1, {|aCampo| nX++, cCampos += aCampo[1] +;  
                    IIF(nX == nMax, ' ', ', ')})  
  
// Monta a query utilizando a String cCampos  
  
cQuery := "SELECT "+cCampos+" FROM "+RetSqlName("SA1")  
  
If Select("SA1SQL") <> 0  
    DbSelectArea("SA1SQL")  
    DbCloseArea()  
Endif  
  
dbUseArea(.T., "TOPCONN", TcGenQry(, , cQuery), "SA1SQL", .T., .F.)
```

**Continuação:**

```

For nX := 1 to Len(aStruSA1)
    IF !( aStrutSA1[nX][2] $ "C/M")
        TCSetField( cAlias,aStruSA1[nX][1],aStruSA1[nX][2],;
                    aStruSA1[nX][3],aStruSA1[nX][4])
    ENDIF
Next nX

```

### **TCSPEEXEC()**

Executa uma Stored Procedure no Banco de Dados.



*Importante*

Devido a uma limitação em alguns dos Bancos de Dados suportados na obtenção dos tipos de parâmetros (se são de INPUT e/ou OUTPUT) todos as Stored Procedures a serem executadas através do TOPConnect deverão obedecer o seguinte padrão de nomenclatura de seus parâmetros :

Parâmetros de INPUT devem começar com IN\_... Ex. IN\_VALOR.

Parâmetros de OUTPUT devem começar com OUT\_... Ex. OUT\_CODIGO

Após a execução de uma Stored Procedure o TOPConnect retornará ao ADVPL um array com 'n' elementos, onde n é o número de parâmetros de OUTPUT da Stored Procedure.

**Sintaxe: TCSPEExec(cSPName,xParam1,xParam2...xParamN)**

**Parâmetros:**

<b>cSPName</b>	Nome completo da Stored Procedure no banco de dados.
<b>xParamN</b>	Parâmetros de input da Stored Procedure.



*Importante*

As procedur es padrões da aplicação ERP Protheus possuem concatenadas em seus nomes o código da empresa para a qual foram aplicadas.

Desta forma ao executar uma procedure padrão do ERP através do comando TCSPEEXEC é necessário utilizar a função xProcedures(), pois a mesma irá concatenar ao nome da procedure que será executada o código da empresa.

Por exemplo: Procedure MAT001

Se for executada com a sintaxe: TCSPEExec("MAT001",...) ocorrerá um erro na aplicação pois esta procedure não existe.

Utilizando a função xProcedures() temos:

TCSPEExec( xProcedures("MAT001"), ...), onde caso a empresa em uso seja a 01, será executada a MAT001\_01, a qual é válida.

**Retorno:**

**Array**

Array com os parâmetros de retorno da Stored Procedure e <cSPName> é o nome da Stored Procedure a ser executada e os demais parâmetros variam conforme a definição da Stored Procedure.

**Exemplo:**

```
//Verifica se a Stored Procedure Teste existe no Servidor
If TCSPEexist("TESTE")

//Executa a Stored Procedure Teste
aRet := TCSPEexec("TESTE", "JOSE", 1000)
if aRet <> nil

    cRetorno := ""
    For i:= 1 to Len(aRet)
        cRetorno += "|" + IIF(ValType(aRet[i]) == "C", aRet[i],;
                               IIF(ValType(aRet[i]) == "N", CValToChar(aRet[i]), "U"))
    Next

    //Mostra os valores de retorno
    ALERT(cRetorno)

Else
    ALERT("Erro executando Stored Procedure"+CRLF+"Mensagem: "+TCSQLError())
Endif
EndIf
```



**Importante**

A função TCSPEexist() deve ser utilizada apenas para verificação da existência de Stored Procedures padrões da aplicação ERP, ou que possuam concatenações no nome a informação da empresa para qual estão habilitadas.

Para verificação de procedures sem esta característica deve ser utilizada a função ExistProc().



**Anotações**

---

---

---

---

## **TCSPEXIST()**

Verifica a existência de uma determinada Stored Procedure no servidor.

- Sintaxe: TCSPEXist(cSPName)**

- Parâmetros:**

<b>CSPName</b>	Nome da procedure a ser verificada no banco de dados.
----------------	---

- Retorno:**

<b>Lógico</b>	Indica se a Stored Procedure existe ou não e <cSPName> é o nome da Stored Procedure procurada.
---------------	--

- Exemplo:**

```
If TCSPEXist("MAT001_01")
    TCSPEExec("MAT001_01",...)
Endif
```



*Importante*

A função TCSPEXist() deve ser utilizada apenas para verificação da existência de Stored Procedures padrões da aplicação ERP, ou que possuam concateadas no nome a informação da empresa para qual estão habilitadas.

Para verificação de procedures sem esta característica deve ser utilizada a função ExistProc().

## **TCSQLError()**

Retorna o último erro registrado pelo TOPConnect durante a execução de uma Query.

- Sintaxe: TCSQLError()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>String</b>	Mensagem de erro registrada no TopConnect para a ocorrência.
---------------	--

- Exemplo:**

```
If TCSQLEexec("UPDATE CUSTOMER SET VALUE=0") < 0 then
    ALERT(TCSQLError())
Endif
```



Importante

Esta é a mesma mensagem que esta registrada no log de eventos do TopConnect / TOTVS DbAccess Manager.

## TCSQLEXEC()

Permite a execução de comandos de atualização no servidor de banco de dados.

- Sintaxe:** **TCSQLEexec(cCommand)**

- Parâmetros:**

<b>cCommand</b>	Comando SQL a ser executado.
-----------------	------------------------------

- Retorno:**

<b>Numérico</b>	Retorna um valor negativo em caso de erros.
-----------------	---

- Exemplo:**

```
nRet := TCSQLEexec("UPDATE CUSTOMER SET VALUE=0")
```

## TCSRVTYPE()

Retorna o tipo do servidor no qual TOPConnect / TOTVS DbAccess está em execução.

- Sintaxe:** **TCSrvType()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>String</b>	Tipo do servidor, como por exemplo: "WinNT" ou "AS/400".
---------------	--

- Exemplo:**

```
TCLink ("MSSQL/TESTE", "TOPSRV")
ALERT(TCSrvtype())
```



Anotações

Devido aos tratamentos específicos necessários nas queries que serão executadas em bancos de dados DB2 com sistemas operacionais AS/400 é normal a seguinte verificação:

```
#IFDEF TOP
    If TcSrvType() != "AS/400"

        <Tratamento SQL convencional>

    Else

#ENDIF

        <Tratamento padrão CODBASE>

#define TOP
    Endif
#endif
```



*Importante*

## **TCUNLINK()**

Encerra uma conexão com o TOPConnect.

- Sintaxe: TCUnlink(nConn)**

- Parâmetros:**

<b>nConn</b>	Número da conexão previamente retornado pela função TCLink()
--------------	--

- Retorno:**

<b>Nenhum</b>	.
---------------	---

- Exemplo:**

```
TCConType("NPIPE")
nConn := TCLink("MSSQL/TOPCONN", "TOPSRV")
TCUnLink(nConn)
```



*Anotações*

---

---

---

---

## **TCCHKOBJ()**

Verifica a existência de um objeto no servidor AS/400

- Sintaxe:** **TCChkObj(cObj,cLibrary,cType)**

- Parâmetros:**

<b>cObj</b>	Nome do objeto a ser verificado.
<b>cLibrary</b>	Nome da biblioteca que deve conter o objeto
<b>cType</b>	Tipo do de objeto AS/400, como por exemplo: *FILE ou *PGM.

- Retorno:**

<b>Numérico</b>	0 quando o objeto existe ou o número do erro no AS/400.
-----------------	---

- Exemplo:**

```
nError := TCChkObj ("CALCCUST", "PRODUCAO", "*PGM")
```



*Importante*

**Função para uso apenas com o TOPConnect em servidores AS/400.**

## **TCEXEERROR()**

Retorna uma string com a mensagem de erro retornada pela execução das funções TCPGMEXE() e TCSYSEX().

- Sintaxe:** **TCExeError()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>String</b>	Mensagem de erro.
---------------	-------------------

- Exemplo:**

```
If TCPGMEXE ("CALCCUST") != 0  
    ALERT (TCExeError ())  
Endif
```



Importante

**Função para uso apenas com o TOPConnect em servidores AS/400.**

## **TCPGMEXE()**

Executa um programa no servidor AS/400.

- Sintaxe: TCPGMEXE(cProgram)**

- Parâmetros:**

cProgram	Nome do programa a ser executado.
----------	-----------------------------------

- Retorno:**

<b>Numérico</b>	0 quando o objeto existe ou o número do erro no AS/400.
-----------------	---

- Exemplo:**

```
If TCPGMEXE ("CALCCUST") != 0  
    ALERT(TCExeError())  
Endif
```



Importante

**Função para uso apenas com o TOPConnect em servidores AS/400.**

## **TCSYSEXEC()**

Permite a execução de um comando utilizando a sintaxe e recursos nativos do DB2/400, diretamente no servidor do banco de dados.

- Sintaxe: TCSysExe(cCommand)**

- Parâmetros:**

<b>cCommand</b>	Comando a ser executado no servidor de banco de dados utilizando sintaxe DB2/400.
-----------------	---

- Retorno:**

<b>Numérico</b>	0 quando o objeto existe ou o número do erro no AS/400.
-----------------	---

**Exemplo:**

```
cCommand := "CRTCBLMOD MODULE ("+cTDataBase+ "/" +cName+"F"+cCrrEmp+) "
cCommand += "SRCFILE ("+cTDataBase+ "/QSPSRC) "
cCommand += "SRCMBR ("+cName+"F"+cCrrEmp+) "
cCommand += "REPLACE (*YES)"

If TCSysExe( cCommand )!= 0
    ALERT(TCExeError())
Endif
```



*Importante*

**Função para uso apenas com o TOPConnect em servidores AS/400.**

### **Funções acessórias para uso em fontes com interação com o TOTVS DbAccess**

#### **CHANGEQUERY()**

Função que efetua as adequações necessárias a query para que a mesma possa ser executada adequadamente no banco de dados em uso pela aplicação ERP através do TopConnect.

Esta função é necessária pois a aplicação ERP Protheus pode ser utilizada com diversos bancos de dados, e cada banco possui particularidades em sua sintaxe, de forma que mesmo uma query escrita respeitando o padrão SQL ANSI pode necessitar de adequações.

**Sintaxe: ChangeQuery(cQuery)**

**Parâmetros:**

<b>cQuery</b>	String contendo a query escrita em sintaxe SQL ANSI.
---------------	--

**Retorno:**

<b>String</b>	Query adequada em função do banco em uso pela conexão ativa com a aplicação TopConnect.
---------------	---

**Exemplo:**

```
cQuery := "SELECT DISTINCT CV8_PROC FROM "+RETSQLNAME("CV8")
cQuery += " WHERE "
cQuery += "CV8_FILIAL = '" +MV_PAR01+ "' AND "
cQuery += "D_E_L_E_T_ = ' '
cQuery += "ORDER BY CV8_PROC"
cQuery := ChangeQuery(cQuery)
dbUseArea(.T., "TOPCONN", TCGenQry(,,cQuery), "CV8QRY", .F., .T.)
```



Importante

A função ChangeQuery() deve obrigatoriamente ser utilizada pelos fontes da aplicação ERP, pois a mesma pode ser instalada em diversos ambientes, utilizando os vários bancos de dados homologados, mas para desenvolvimentos específicos de clientes, o analista pode optar por não utilizar a função ChangeQuery().

A razão disto é permitir ao analista, que tenha conhecimentos mais apurados do banco de dados em uso pelo cliente, utilizar uma sintaxe focada em utilizar melhor os recursos do banco, otimizando a performance da aplicação.

## **RETFULLNAME()**

Retorna o nome real da tabela no banco de dados para utilização desta na seleção da query.



Importante

A função RetFullName() não é exclusiva para uso em queries, sendo utilizada pela aplicação ERP para verificar o nome da tabela para o alias, especificado no SX2 da empresa indicada.

**Sintaxe: RetFullName(cAlias, cEmp)**

**Parâmetros:**

<b>cAlias</b>	Nome da área de trabalho da aplicação para identificação da tabela no banco de dados.
<b>cEmp</b>	Código da empresa a ser avaliada.

**Retorno:**

<b>String</b>	Nome completo / real da tabela no banco de dados.
---------------	---

**Exemplo:**

```
SA1->(DbSetOrder(1)) // A1_FILIAL+A1_COD+A1_LOJA
cOrder := SqlOrder(IndexKey())

cQuery := "SELECT * FROM "+RETSQLNAME("SA1")+" WHERE "
cQuery += RetSqlCond("SA1")
cQuery += " ORDER BY "+cOrder

dbUseArea(.T.,"TOPCONN", TcGenQry(,cQuery), "TRBSQL", .T., .F.)

cRealName := RetFullName("TRBSQL")
```

## **RETSQLCOND()**

---

Retorna uma string com as condições padrões de uso em uma query.

**Sintaxe:** **RetSqlCond(cAlias)**

**Parâmetros:**

<b>cAlias</b>	Nome do alias para composição a string de condições.
---------------	--

**Retorno:**

<b>String</b>	Condições de seleção padrões para o alias indicado. As condições padrões são: ALIAS_FILIAL = xFilial(ALIAS) .AND. ALIAS.D_E_L_E_T_ = "".
---------------	---

**Exemplo:**

```
cQuery := "SELECT DISTINCT CV8_PROC FROM "+RETSQLNAME("CV8")
cQuery += " WHERE "
cQuery += RetSqlCond("CV8")
cQuery += "ORDER BY CV8_PROC"
cQuery := ChangeQuery(cQuery)
dbUseArea(.T., "TOPCONN", TCGenQry(, , cQuery), "CV8QRY", .F., .T.)
```

---

## **RETSQLNAME()**

Retorna o nome padrão da tabela para seleção no banco de dados através da query.

**Sintaxe:** **RetSqlName(cAlias)**

**Parâmetros:**

<b>cAlias</b>	Alias para ser avaliado o nome padrão da tabela.
---------------	--

**Retorno:**

<b>String</b>	Nome completo da tabela para seleção através da query.
---------------	--

**Exemplo:**

```
cQuery := "SELECT DISTINCT CV8_PROC FROM "+RETSQLNAME("CV8")
cQuery += " WHERE "
cQuery += RetSqlCond("CV8")
cQuery += "ORDER BY CV8_PROC"
cQuery := ChangeQuery(cQuery)
dbUseArea(.T., "TOPCONN", TCGenQry(, , cQuery), "CV8QRY", .F., .T.)
```

## **RETSQTABLE()**

Retorna o nome real da tabela para seleção no banco de dados através da query.

- Sintaxe:** RetSqlTable(cAlias)

- Parâmetros:**

<b>cAlias</b>	Alias para ser avaliado o nome real da tabela.
---------------	--

- Retorno:**

<b>String</b>	Nome real da tabela no banco de dados.
---------------	--

- Exemplo:**

```
cQuery := "SELECT DISTINCT CV8_PROC FROM "+RETSQTABLE("CV8")
cQuery += " WHERE "
cQuery += RetSqlCond("CV8")
cQuery += "ORDER BY CV8_PROC"
cQuery := ChangeQuery(cQuery)
dbUseArea(.T., "TOPCONN", TCGenQry(, , cQuery), "CV8QRY", .F., .T.)
```

## **SQLCOPY()**

Cria um arquivo no formato especificado pela configuração LOCALFILES do ambiente com o retorno da query.

- Sintaxe:** SqICopy(cFile, cWhere, aStru, cAlias, aDates, IRecno)

- Parâmetros:**

<b>cFile</b>	Nome do arquivo temporario destino a ser gerado
<b>cWhere</b>	Condicao "Where" da query
<b>aStru</b>	Array com os Campos a serem selecionados pela query
<b>cAlias</b>	Alias origem dos dados
<b>aDates</b>	Array com os nomes dos campos que deverão ter o tratamento da função TCSetField() para o formato de data.
<b>IRecno</b>	Se copia o conteúdo da coluna R_E_C_N_O_ para o arquivo gerado.

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **SQLORDER()**

---

Converte a sintaxe de um índice em formato ISAM (CODEBASE) para o formato SQL ANSI

- Sintaxe: SqlOrdem(cOrder)**

- Parâmetros:**

<b>cOrder</b>	Sintaxe de índice em formato ISAM.
---------------	------------------------------------

- Retorno:**

<b>String</b>	Índice convertido para sintaxe SQL ANSI
---------------	---

- Exemplo:**

```
SA1->(DbSetOrder(1)) // A1_FILIAL+A1_COD+A1_LOJA
cOrder := SqlOrder(IndexKey())

cQuery := "SELECT * FROM "+RETSQLNAME("SA1")+" WHERE "
cQuery += RetSqlCond("SA1")
cQuery += " ORDER BY "+cOrder

dbUseArea(.T.,"TOPCONN", TcGenQry(,,cQuery), "TRBSQL", .T., .F.)

cRealName := RetFullName("TRBSQL")
```



*Anotações*

---

---

---

---

## **SQLTOTRB()**

Preenche um arquivo temporário com o conteúdo do retorno da query.

- Sintaxe: SqlToTrb(cQuery, aStru, cAlias)**

- Parâmetros:**

<b>cQuery</b>	Query para seleção dos registros no banco de dados.
<b>aStru</b>	Array contendo a estrutura do arquivo temporário a ser preenchido com os dados de retorno da query.
<b>cAlias</b>	Alias do arquivo temporário a ser preenchido com os dados de retorno da query.



*Importante*

Para correta utilização da função SQLTOTRB() o arquivo temporário deve estar criado e com a área de trabalho (ALIAS) definido.

- Retorno:**

<b>Nenhum</b>	.
---------------	---

- Exemplo:**

```
// Arquivo de trabalho que será utilizado pela MaWndBrowse

cCampos:= ""
nX      := 0
nMax    := Len(aStruSQL)

aEval( aStruSQL,{|aCampo| nX++, cCampos += aCampo[1] +;
                  IIF(nX == nMax, ' ', ',' )} )

cArqTrb := CriaTrab(aStruSQL,.T.)
dbUseArea(.T.,__LOCALDRIVER,cArqTrb,cAlias,.T.,.F.)
cQuery := "SELECT "+cCampos+" FROM "+RetSqlName("SA1")+" (NOLOCK) "
SqlToTrb(cQuery,aStruSQL,cAlias)

cArqInd := CriaTrab(Nil,.F.)
cChave   := "A1_FILIAL+A1_COD+A1_LOJA"
IndRegua(cAlias,cArqInd,cChave,,, "Indexando Registros...")

dbSelectArea( cAlias )
dbGotop()
```

## **8.3. Aplicações com controle de comunicação com o Banco de Dados**

Utilizando as função de interação com a aplicação TopConnect / TOTVS DbAccess têm-se um grande número de aplicações que podem ser desenvolvidas, combinando-se recursos previamente estudados nos módulos Básico, Intermediário e utilizando os conceitos de orientação à objetos e as classes de objetos disponíveis na linguagem ADVPL, tais como:

- Rotinas de processamento utilizando queries para seleção dos registros;
- Relatórios para seleção dos registros;
- Rotinas de atualização para execução dos comando de atualização diretamente no banco de dados.

Em resumo, qualquer aplicação do ambiente ERP pode ser escrita utilizando os recursos de interação com a aplicação TopConnect / TOTVS DbAccess, para obter, manipular e atualizar os dados do banco de dados.

Neste tópico serão detalhadas duas aplicações utilizando estas funções, cujo grau de complexidade ilustra bem o potencial desta ferramenta.

- MaWndBrowse com Alias Temporário gerado por Query**
- Banco de dados de interface**

### **8.3.1. MaWndBrowse com Alias Temporário gerado por Query**

Conforme descrito no tópico 6.1. MaWndBrowse, esta função permite a montagem de um browse cuja estrutura do arquivo não necessita estar definida no dicionário de dados da aplicação ERP.

Utilizando esta particularidade da MaWndBrowse é possível então compor um alias temporário contendo o retorno de uma query e exibi-lo em um browse.

A vantagem deste “BrowseSQL” em relação ao browse de uma tabela “normal” é a possibilidade de combinar informações de várias tabelas através de uma query e exibi-las em tela.

Algumas aplicações interessantes para este recurso seriam:

- Consulta Kardex em tela, pois este relatório combina dados das tabelas de itens dos documentos de entrada (SD1), itens dos documentos de saída (SD2) e movimentações internas (SD3).
- Conciliação de contabilizações, pois uma contabilização é composta por uma origem, a qual pode ser praticamente de qualquer tabela de movimentos da aplicação ERP, e de um ou mais lançamentos contábeis (CT2) gerados por esta origem.
- Demonstração em tela das depreciações dos bens do imobilizado por período, compondo dinamicamente as colunas de depreciação de acordo com o intervalo de meses selecionados. Este recurso é interessante pois elimina a limitação de espaço da impressão e pode ser combinado com a funcionalidade de exportação de grids de dados para o Microsoft Excel.

## **Exemplo: MaWndBrowse com Alias Temporário gerado por Query**



Este exemplo de uso da MaWndBrowse com alias de temporário utilizando queries é derivado do exemplo de utilização da MaWndBrowse com arquivo temporário descrito no item 6.1. MaWndBrowse.

```
#include "protheus.ch"

/*
+-----+
| Função | WndSQLTRB | Autor | Arnaldo R. Junior | Data | |
+-----+
| Descrição | Demonstra a utilização da MaWndBrowse com SQL | |
+-----+
| Uso | Curso ADVPL | |
+-----+
*/

User Function WndSQLTRB()

// Variáveis para o Arquivo Temporario
Local cChave      := ""
Local cArqTrb     := ""
Local aStruTRB    := {}
Local aStruSQL    := SA1->(DbStruct())

// Variáveis para o MaWndBrowse
Local cTitulo      := "Cadastro Temporario"// Título obrigatório
Local cAlias       := "SA1" // Alias da tabela corrente podendo ser TRB
Local cFunLeg     := "" // Função que deverá retornar um valor lógico e com isso
será atribuído semafóro na primeira coluna do browse
Local cTopFun     := "" // Mostrar os registros com a chave de
Local cBotFun     := "" // Mostrar os registros com a chave ate
Local lCentered   := .T. // Valor verdadeiro centraliza
Local aResource   := {} // aAdd(aResource,{"IMAGEM","Texto significativo"})
Local nModelo     := 1 // 1- Menu do aRotina
Local aPesqui     := {} // aAdd(aPesqui{"Título",nOrdem}), se não passado será
utilizado o AxPesqui
Local cSeek       := "" // Chave principal para a busca, exemplo: xFilial("??")
Local lDic        := .T. // Parâmetro em conjunto com aCampos
Local lSavOrd    := .T. // Estabelecer a ordem após pesquisas.

// Variaveis para a MsAdvSize
Local lEnchBar    := .F. // Se a janela de diálogo possuirá enchoicebar (.T.)
Local lPadrao     := .F. // Se a janela deve respeitar as medidas padrões do
Protheus (.T.) ou usar o máximo disponível (.F.)
Local nMinY       := 400 // Altura mínima da janela

Local aSize       := MsAdvSize(lEnchBar, lPadrao, nMinY)

Private cCadastro := " "
Private aCampos   := {} // Se lDic=.T. utilizará o SX3, do contrário o aCampos
informado -> aAdd(aCampo,{X3_CAMPO,X3_PICTURE,X3_TITULO,X3_TAMANHO})
Private aRotina   := {} // Idêntico ao aRotina para mBrowse
```

**Continuação:**

```
aAdd(aRotina, {"Visualizar", "U_TcVisual", 0, 2}) // Desenvolver Enchoice para
campos de arquivo temporário
aAdd(aRotina, {"Incluir" , "U_TcInclui", 0, 3}) // Desenvolver Enchoice para
campos de arquivo temporário
aAdd(aRotina, {"Alterar" , "U_TcAltera", 0, 4}) // Desenvolver Enchoice para
campos de arquivo temporário

aAdd(aRotina, {"Excluir" , "U_TcExclui", 0, 5}) // Desenvolver Enchoice para
campos de arquivo temporário

// Estrutura do Arquivo: Nome do campo / tipo, tamanho, decimais (SX3 para
temporário)
/*
AADD(aStruTRB, {"TRB_FILIAL" , "C", 02, 0})
// Nome_Campo , Tipo_Campo, Tamanho, Decimal
AADD(aStruTRB, {"TRB_ID" , "C", 14, 0})
AADD(aStruTRB, {"TRB_NOME" , "C", 20, 0})
AADD(aStruTRB, {"TRB_IDADE" , "N", 03, 0})
AADD(aStruTRB, {"TRB_STATUS" , "C", 01, 0})
*/
// aCampos padrão para a MaWndBrowse
//AADD(aCampos, {<Nome_Campo>, <Picture>, <Titulo>, <Tamanho>})

// aCampos melhorado para a WndBrwTRB
//AADD(aCampos, {<Nome_Campo>, <Picture>, <Titulo>, <Tamanho>, <Tipo>, <cWhen>, ;
//<lObrigatorio>})
// Nota: lObrigatorio deve ser sempre a ultima informacao do aCampos
/*
AADD(aCampos, {"TRB_FILIAL" , "@!" , "Filial" , 02, "C", ".F.", .T.})
AADD(aCampos, {"TRB_ID" , "@!" , "Matricula" , 14, "C", ".F.", .T.})
AADD(aCampos, {"TRB_NOME" , "@!" , "Nome" , 20, "C", ".T.", .F.})
AADD(aCampos, {"TRB_IDADE" , "@E 999", "Idade" , 03, "N", ".T.", .F.})
AADD(aCampos, {"TRB_STATUS" , "@!" , "Status" , 01, "C", ".T.", .T.})
*/
If ( Select( cAlias ) <> 0 )
    dbSelectArea ( cAlias )
    dbCloseArea ()
Endif

// Arquivo de trabalho que será utilizado pela MaWndBrowse

cCampos:= ""
nX      := 0
nMax   := Len(aStruSQL)
aEval( aStruSQL, {|aCampo| nX++, cCampos += aCampo[1] +;
    IIF(nX == nMax, ' ', ', ')})

cArqTrb := CriaTrab(aStruSQL,.T.)
dbUseArea(.T.,__LOCALDRIVER,cArqTrb,cAlias,.T.,.F.)
cQuery := "SELECT "+cCampos+" FROM "+RetSqlName("SA1")+" (NOLOCK) "
SqlToTrb(cQuery,aStruSQL,cAlias)
```

**Continuação:**

```
cArqInd := CriaTrab(Nil,.F.)
cChave      := "A1_FILIAL+A1_COD+A1_LOJA"

IndRegua(cAlias,cArqInd,cChave,,, "Indexando Registros...")

dbSelectArea( cAlias )
dbGotoP()

MaWndBrowse(aSize[7],aSize[2],aSize[6],aSize[5],cTitulo,cAlias,/*aCampos*/,;
            aRotina,,cTopFun,cBotFun,lCentered,,nModelo,,cSeek,lDic,lSavOrd)

If ( Select( cAlias ) <> 0 )
    dbSelectArea ( cAlias )
    dbCloseArea ()
Endif

If File(cArqInd+OrdBagExt())
    FErase(cArqInd+OrdBagExt())
ENDIF

DbSelectArea("SA1")
DbSetOrder(1)

Return

/*
+-----
| Função      | TcVisual           | Autor | Arnaldo R. Junior | Data | |
+-----
| Descrição   | Enchoice para arquivos temporarios | |
+-----
| Uso         | Curso ADVPL          | |
+-----
*/
USER FUNCTION TcVisual(cAlias,nReg,nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcao de verdade
LOCAL bOk         := {||oDlg:End()}
LOCAL bCancel     := {||oDlg:End()}
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>"      , {nLinha,nColuna}
// , "<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>","<Tamanho>"})
// aCampos, {"TRB_FILIAL"           , "@!" , "Filial" , 02}
```

**Continuação:**

```

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 ,nLinha +015,nLinha) // Impar
        nColuna := IIF(nX%2 == 0,nColuna := nColuna+170,10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1],{nLinha,nColuna} ,
                      aCampos[nX][3],aCampos[nX][2],"AllwaysTrue()",;
                      "",".F.",aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    & ("_" +aCampos[nX][1]) := (cAlias)->&(aCampos[nX][1])

Next nX

oDlg := TDialoG():New(000,000,400,650,cCadastro,,,.T.)

TEnchoice(oDlg, aCposEnch,,,.T.)

oDlg:bInit := {|| EnchoiceBar(oDlg, bOk, bCancel,.F.,{},nReg,cAlias)}
oDlg:lCentered := .T.
oDlg:Activate()

RETURN

/*
+-----
| Função | TcInclui | Autor | Arnaldo R. Junior | Data | |
+-----
| Descrição | Enchoice para arquivos temporarios | |
+-----
| Uso | Curso ADVPL | |
+-----
*/
USER FUNCTION TcInclui(cAlias,nReg,nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk
LOCAL bCancel     := {||oDlg:End()}
LOCAL aArea       := GetArea()
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>" ,{nLinha,nColuna}
// ,,<Titulo>,"<Picture>","<Validacao>","<F3>","<lWhen>,<Tamanho>})
// aCampos,{"TRB_FILIAL" , "@!" , "Filial" ,02}

```

**Continuação:**

```

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 ,nLinha := nLinha +015,nLinha) // Impar
        nColuna := IIF(nX%2 == 0,nColuna := nColuna+170,10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1],{nLinha,nColuna} ,;
                      aCampos[nX][3],aCampos[nX][2],"AllwaysTrue()",;
                      "",".T.",aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

Do Case
    Case aCampos[nX][5] == "C"
        & ("_" +aCampos[nX][1]) := Space(aCampos[nX][4])
    Case aCampos[nX][5] == "N"
        & ("_" +aCampos[nX][1]) := 0
    Case aCampos[nX][5] == "D"
        & ("_" +aCampos[nX][1]) := CTOD("")
    Case aCampos[nX][5] == "L"
        & ("_" +aCampos[nX][1]) := .F.
    Case aCampos[nX][5] == "M"
        & ("_" +aCampos[nX][1]) := Space(aCampos[nX][4])
EndCase

Next nX

oDlg := TDIALOG():New(000,000,400,650,cCadastro,,,.T.)

TEnchoice(oDlg, aCposEnch,,,.T.)

bOk := {|| IIF( U_TValid(cAlias,nReg,nOpcE,aCampos),;
                  ( U_TGravar(cAlias,nReg,nOpcE,aCampos),oDlg:End()),) }

oDlg:bInit := {|| EnchoiceBar(oDlg, bOk, bCancel,.F.,{},nReg,cAlias) }
oDlg:lCentered := .T.
oDlg:Activate()

RETURN

```



### Anotações

---



---



---



---



---

**Continuação:**

```
/*
+-----+
| Função      | TcAltera          | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição   | Enchoice para arquivos temporarios           |
+-----+
| Uso         | Curso ADVPL          |           |
+-----+
/*

USER FUNCTION TcAltera(cAlias,nReg,nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk
LOCAL bCancel    := {||oDlg:End()}
LOCAL aArea       := GetArea()
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>"      ,{nLinha,nColuna}
// ,,<Titulo>,"<Picture>","<Validacao>","<F3>","<lWhen>",<Tamanho>})
// aCampos, {"TRB_FILIAL"           , "@!" , "Filial"      ,02}

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 ,nLinha := nLinha +015,nLinha) // Impar
        nColuna := IIF(nX%2 == 0 ,nColuna := nColuna+170,10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1],{nLinha,nColuna} ,
                    aCampos[nX][3],aCampos[nX][2],"AlwaysTrue()",;
                    "",aCampos[nX][6],aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    &("_" +aCampos[nX][1]) := (cAlias)->&(aCampos[nX][1])

Next nX

oDlg := TDIALOG():New(000,000,400,650,cCadastro,,,.T.)

TEnchoice(oDlg, aCposEnch,,,.T.)

bOk := {|| IIF( U_TValid(cAlias,nReg,nOpcE,aCampos),;
               ( U_TGravar(cAlias,nReg,nOpcE,aCampos),oDlg:End()),) }

oDlg:bInit := {|| EnchoiceBar(oDlg, bOk, bCancel,.F.,{},nReg,cAlias)}
oDlg:lCentered := .T.
oDlg:Activate()

RETURN
```

**Continuação:**

```
/*
+-----+
| Função      | TcExclui          | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição   | Enchoice para arquivos temporarios |
+-----+
| Uso         | Curso ADVPL           |
+-----+
*/
USER FUNCTION TcExclui(cAlias, nReg, nOpc)

LOCAL aCposEnch := {}
LOCAL nLinha      := 15
LOCAL nColuna     := 10
LOCAL nOpcE       := aRotina[nOpc][4] // Opcão de verdade
LOCAL bOk
LOCAL bCancel    := {||oDlg:End()}
LOCAL nX

// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aCposEnch, {"<Variavel>"      ,{nLinha,nColuna}
// ,<Titulo>,"<Picture>","<Validacao>","<F3>","<lWhen>,<Tamanho>})
// aCampos,{ "TRB_FILIAL"           , "@!" , "Filial"        ,02}

For nX := 1 to Len(aCampos)

    If nX > 1
        nLinha := IIF(nX%2 > 0 ,nLinha := nLinha +015,nLinha) // Impar
        nColuna := IIF(nX%2 == 0,nColuna := nColuna+170,10)// Par
    Endif

    AADD(aCposEnch, {"_" +aCampos[nX][1],{nLinha,nColuna} ,
                    aCampos[nX][3],aCampos[nX][2],"AllwaysTrue()",;
                    "", ".F.",aCampos[nX][4]})

    SetPrvt("_" +aCampos[nX][1])

    &("_" +aCampos[nX][1]) := (cAlias)->&(aCampos[nX][1])

Next nX

oDlg := TDIALOG():New(000,000,400,650,cCadastro,,,.T.)

TEnchoice(oDlg, aCposEnch,,,.T.)

bOk := {|| IIF( U_TValid(cAlias,nReg,nOpcE,aCampos),;
                (U_TGravar(cAlias,nReg,nOpcE,aCampos),oDlg:End()),) }

oDlg:bInit := {|| EnchoiceBar(oDlg, bOk, bCancel,.F.,{},nReg,cAlias)}
oDlg:lCentered := .T.
oDlg:Activate()

RETURN
```

**Continuação:**

```

/*
+-----+
| Função      | TcValid           | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição   | Enchoice para arquivos temporarios |
+-----+
| Uso         | Curso ADVPL        |
+-----+
*/

USER FUNCTION TcValid(cAlias,nReg,nOpc,aCampos)
LOCAL lRet
LOCAL nX
LOCAL nPosObrig := Len(aCampos[1])

For nX := 1 to Len(aCampos)
    IF aCampos[nX,nPosObrig] == .T.
        IF !(lRet := !Empty(&("_"+aCampos[nX,1])))
            Help("TEnchoice",1,"HELP","OBRIGATORIO","Existem campos
obrigatorios nao preenchidos",1,0)
            RETURN lRet // EXIT
        ENDIF
    ENDIF
Next nX

IF nOpc == 3
    IF !(lRet := !((cAlias)->(dbSeek(_TRB_FILIAL+_TRB_ID))))
        Help("TEnchoice",1,"HELP","INCLUSAO","Ja existe um registro com esta
chave",1,0)
        ENDIF
ELSE
    IF !(lRet := (cAlias)->(dbSeek(_TRB_FILIAL+_TRB_ID)))
        Help("TEnchoice",1,"HELP","ALTERACAO","Nao existe um registro com
esta chave",1,0)
        ENDIF
ENDIF

RETURN lRet

```



### Anotações

---



---



---



---



---

**Continuação:**

```
/*
+-----+
| Função      | TcGravar          | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição   | Enchoice para arquivos temporarios |
+-----+
| Uso         | Curso ADVPL          |
+-----+
/*

USER FUNCTION TcGravar(cAlias,nReg,nOpc,aCampos)
LOCAL nX

RecLock(cAlias,nOpc==3)
    IF nOpc == 5
        DbDelete()
    ELSE
        For nX := 1 to Len(aCampos)
            (cAlias)->&(aCampos[nX][1]) := &("_"+aCampos[nX][1])
        Next nX
    ENDIF
Msunlock()

RETURN
```



**Anotações**

---

---

---

---

**Continuação:**

```
/*
+-----+
| Função      | TEnchoice      | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição  | Enchoice para arquivos temporarios          |
+-----+
| Uso        | Curso ADVPL                         |
+-----+
*/

Static Function TEnchoice(oDlg, aCampos, nLeftE, nTopE, nHeightE, nWidthE,;
lEnchBar)

Local aSays      := {}
Local aGets      := {}
Local cCaption   := ""
Local cPict      := ""
Local cValid     := ""
Local cF3        := ""
Local cWhen      := ""
Local cBlKSay    := ""
Local cBlkGet    := ""
Local cBlKVld   := ""
Local cBlKWhen   := ""
Local nLeft      := 0
Local nTop       := 0
Local nI         := 0

Default lEnchBar := .F.
Default   nLeftE           := IIF(lEnchBar,(oDlg:nLeft)+16,2)
Default   nTopE            := 2
Default   nHeightE         := (oDlg:nHeight)-135
Default   nWidthE          := (oDlg:nWidth)-102
// Padrao Modelo2(), com lWhen -> cWhen
// AADD(aC, {"<Variavel>", {nLinha,nColuna}
//,"<Titulo>","<Picture>","<Validacao>","<F3>","<lWhen>",<Tamanho>})

If Len(aCampos) > 0

    oScroll := TScrollBox():New(oDlg, nLeftE, nTopE, nHeightE , nWidthE, .T.,;
                                .T., .T.)

    For nI:=1 to Len(aCampos)

        If Len(aCampos[nI])==8

            cCampo      := aCampos[nI,1]
            nLeft := aCampos[nI,2,1]-13
            nTop  := aCampos[nI,2,2]
            cCaption:= Iif(Empty(aCampos[nI,3]), " " ,aCampos[nI,3])
            cPict   := Iif(Empty(aCampos[nI,4]), Nil ,aCampos[nI,4])
            cValid  := Iif(Empty(aCampos[nI,5]), ".t." ,aCampos[nI,5])
            cF3     := Iif(Empty(aCampos[nI,6]), NIL ,aCampos[nI,6])

            cWhen := Iif(Empty(aCampos[nI,7]), ".T." ,aCampos[nI,7])
```

**Continuação:**

```
nWidthG      := Iif(Empty(aCampos[nI,8]), 100,;
                     Iif(aCampos[nI,8]*3.5 > 100,100,nil))

cBlkSay := " { || OemToAnsi('"+cCaption+"') } "
cBlkGet := " { | u | If( PCount() == 0,;
                     "+cCampo+", "+cCampo+":= u ) } "
cBlkVld := " { || "+cValid+" } "
cBlkWhen := " { || "+cWhen+" } "

AADD(aSays,Array(1))
aSays[nI] := TSay():New(nLeft+1, nTop, &(cBlkSay), oScroll,,,;
                        .F., .F., .F., .T.,,, 50, 8, .F., .F., .F.,;
                        .F. )
AADD(aGets,Array(1))

aGets[nI] := TGet():New( nLeft, nTop+50, &(cBlkGet), oScroll,;
                        nWidthG, 11, cPict, &(cBlkVld),,,,.F.,, .T.,,,;
                        .F., &(cBlkWhen), .F., .F.,, .F., .F., cF3,;
                        (cCampo))

EndIf
Next
Endif

Return
```



**Anotações**

---

---

---

---

### 8.3.2. Banco de dados de interface

O termo “Banco de dados de interface” é utilizado para classificar uma aplicação ADVPL escrita com funções de interação com a aplicação TopConnect / TOTVS DbAccess para realizar a integração entre a aplicação ERP Protheus e outro sistema, utilizando como meio de comunicação um banco de dados e não mais importação de arquivos em “N” formatos, tais como .TXT, .DBF e similares.



Importante

Este tipo de integração não visa substituir formatos de integrações padrões já existentes como o padrão CNAB e o padrão Web XML.  
Sua aplicação é integração direta entre sistemas através de um banco de dados com formato e padrões comuns entre elas.

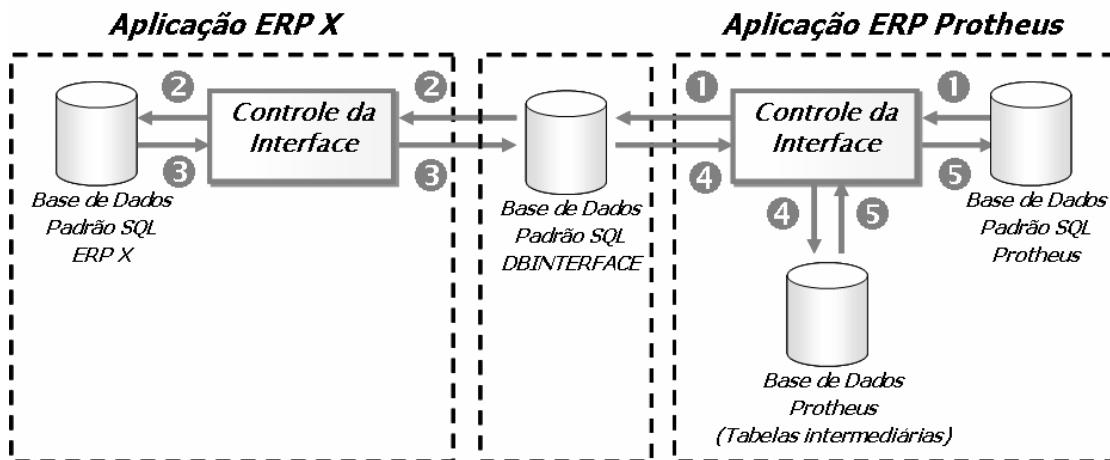
As vantagens da utilização do conceito de Banco de Dados de Interface em comparação da integração por importação / exportação de arquivos são:

- Maior controle da integração das informações entre os sistemas, permitindo um tratamento adequado de eventos de erros;
- Transparência e rastreabilidade das informações integradas, implementando confiabilidade no processo;
- Possibilidade de utilização de um tipo de banco de dados pela aplicação ERP Protheus e de outro tipo de banco de dados para integração entre os sistemas.
- Possibilidade de desenvolvimento de ferramentas para monitorar e gerenciar a integração entre os sistemas através do controle da área de interface no banco de dados, inclusive com a utilização de Stored Procedures para aumentar a performance da aplicação de interface.

#### **Considerações relevantes sobre as funções TCLink() e TCSetConn()**

- A função TCLink() permite a abertura de uma conexão com um banco de dados, desde que haja o OBDC configurado no ambiente da aplicação e na aplicação TOPConnect / TOTVS DBAccess;
- Cada abertura de conexão através da função TcLink() consome uma licença de usuário do TOPConnect / TOTVS DBAccess. Após a abertura de uma conexão a mesma deve ser selecionada pela função TcSetConn() para se tornar a conexão ativa;
- Podem ser utilizadas diversas conexões em uma mesma aplicação ADVPL, e podem ser utilizadas as informações das tabelas já abertas e vinculadas a uma área de trabalho independente da conexão que está ativa. Somente deve ser verificado que para abertura de uma tabela e vínculo desta com uma área de trabalho obrigatoriamente deve estar ativa a conexão com o banco de dados que contém esta tabela.
- A conexão aberta entre o ERP Protheus e o TopConnect / DbAccess possui o número de referência 0. Desta forma, após finalizar uma conexão aberta com o comando TCLink() deve ser efetuado um TCSetConn(0) para restaurar como conexão ativa a conexão padrão do Protheus com o TopConnec.

A figura abaixo ilustra a estrutura típica de uma aplicação utilizando o conceito de Banco de Dados de Interface, com foco na aplicação ERP Protheus e seus recursos pré-disponíveis:



Tarefa	Descrição	Detalhamento
<b>01</b>	<b>Exportação Protheus -&gt; DbInterface</b>	Utilização de queries ou stored procedures para atualização das tabelas do banco DBInterface no formato necessário ao ERP X, executadas a partir do ERP Protheus.
<b>02</b>	<b>Importação DbInterface -&gt; ERP X</b>	Ferramentas proprietárias do ERP X para leitura do DBInterface e tratamento das informações para atualização do banco de dados desta aplicação.
<b>03</b>	<b>Extração ERP X -&gt; DbInterface</b>	Pode ser uma ferramenta proprietária da aplicação ERP X, ou pode ser um conjunto de queries e stored procedures desenvolvidas em conjunto com a equipe do ERP X e a equipe TOTVS, permitindo o controle de execução da extração pelo ERP Protheus.
<b>04</b>	<b>Importação DbInterface -&gt; Protheus</b>	Utilização de queries ou stored procedures para leitura das informações do DBInterface e atualização de tabelas intermediárias no banco de dados do ERP Protheus, executadas a partir do ERP Protheus.
<b>05</b>	<b>Efetivação Protheus -&gt; Protheus</b>	Utilização de funções próprias para esta finalidade no ERP Protheus para efetivação das informações importadas em informações válidas no ERP. Estas funções normalmente são compatíveis com a execução através do recurso de MsExecAuto().

## **Considerações complementares sobre o conceito de Banco de Dados de Interface**

---

- A utilização de tabelas intermediárias no banco de dados Protheus apesar de não ser obrigatória é recomendada para permitir um tratamento adequado das ocorrências de efetivação das informações, principalmente quando utilizada a função MsExecAuto() para execução das rotinas de efetivação dos dados;
- Para extração das informações tanto do banco de dados Protheus para exportação, quanto do banco de dados do sistema ERP X é recomendável o uso de Stored Procedures para aumentar a performance do processo como um todo. É importante lembrar que se o sistema ERP X possuir procedures de exportação as mesmas poderão ser executadas diretamente pelo Protheus através da função TCSpExec();
- É recomendável que as funcionalidades de Exportação, Extração, Importação e Efetivação sejam tratadas tanto com interfaces de execução direta quanto através de execução via Job.



*Anotações*

---

---

---

---

---

## **8.4. Embedded SQL – Facilitador de Query's**

O objetivo do Embedded SQL é facilitar a escrita e leitura de query's. Foi definida uma sintaxe para que se possa escrever a query diretamente no código ADVPL, sem a necessidade de ficar concatenando pedaços de string para compor a string final.

### **Disponibilidade do Recurso**

Este recurso está disponível apenas no Protheus 8. A utilização do Embedded SQL divide-se em : compilação do fonte, e execução do fonte. Para ser possível compilar um fonte com o código escrito no formato Embedded, deve ser utilizado um Protheus, com Build igual ou superior a 7.00.050721p, em um ambiente com repositório para TopConnect ( RPODB=TOP ).

A utiliação deste recurso também depende da atualização da LIB 811, superior a 'Lib 20050902 - 811'.

#### **Exemplo básico - Fonte atual**

```
cQuery : 'SELECT SE2.E2_PREFIXO,SE2.E2_NUM '
cQuery += 'FROM '+RetSqlTable('SE2')+' SE2,'+RetSqlTable('QEK')+ ' QEK '
cQuery += 'WHERE SE2.E2_FILIAL= '+xfilial('SE2')+ ' AND '
cQuery += 'SE2.E2_PREFIXO<> '''+cPrefixo+'' AND '
cQuery += 'SE2.D_E_L_E_T_ = ' ''
cQuery += 'ORDER BY '+RetSqlOrder('SE2')
dbUseArea(.T.,'TOPCONN',TcGenQry(,,cQuery),'E2TEMP',.T.,.T.)
TCSetField('E2TEMP','E2_EMISSAO','D',8,0)
```

#### **Exemplo básico - Utilizando Embedded SQL**

```
BeginSql alias 'E2TEMP'
    column E2_EMISSAO as Date
    %noparser%
    SELECT SE2.E2_PREFIXO,SE2.E2_NUM
    FROM %table:SE2% SE2,%table:QEK% QEK
    WHERE SE2.E2_FILIAL= %xfilial:SE2% AND
        SE2.E2_PREFIXO<> %exp:cPrefixo% AND
        SE2.%notDel%
    ORDER BY %Order:SE2%
EndSql
```

## **Características operacionais - Sintaxe**

---

O bloco onde será escrito o Select deve sempre ser iniciado com '**BeginSql alias**' e finalizado com '**EndSql**'.

Partes do código que devem ser substituídas aparecem entre os sinais de %. Estas expressões possuem tratamento especial em momento de execução.

Qualquer instrução colocada entre BEGINSQL... ENDSQL, que não seja uma expressão %...% , será inserida na query a ser enviada para o banco, de forma literal.

Variáveis, expressões, funções aparecem iniciando com **%exp: %**.

Em **column**, especificar campos da query que são do tipo data, lógico ou numérico (DATE, LOGIC, NUMBER). Esta linha é trocada por chamadas à função **TCSetField**.

**%noparser%** indica que a query não deve passar pela função '**ChangeQuery**' antes de ser enviada ao banco de dados. Caso não especificado, o default é a string da query ser passada automaticamente pela função ChangeQuery.

**%table:<alias>%** é substituída por **RetSqlName(<alias>)**

**%notDel%** é substituída por **D\_E\_L\_E\_T\_= ''**

**%Order:<alias>%** é substituída por **SqlOrder(<alias>->(IndexKey()))**

Há 3 opções para o %Order:

1. **%Order: <cAlias> %** traduzido para  
SqlOrder(<cAlias>->(IndexKey()))
2. **%Order: <cAlias>, <nIndice>%** traduzido para  
SqlOrder(<cAlias>->(IndexKey(<nIndice>)))
3. **%Order: <cAlias>, <cNick>%** traduzido para  
SqlOrder(<alias>->(DBNickIndexKey(<cNick>)))

### **Limitação:**

---

Não é permitido incluir funções no meio do código 'embedded'. Se precisar, o valor deve ser guardado em uma variável antes do início do BeginSql.

```
tam_cp := GetE2ValorSize()

BeginSql alias 'E2TEMP'
column E2_EMISSAO as Date, E2_VALOR as Numeric(tam_cp,2)
. .
EndSql
```

## **Erros de Compilação**

Caso seja utilizado algum argumento inválido para especificar as colunas, ou erros de sintaxe nas expressões a serem transformadas para a montagem da query, a compilação do fonte é interrompida com a ocorrência 'Syntax Error', informando a linha onde a primeira ocorrência foi encontrada.

```
ENDSQL (Error C2001 Syntax error:)
```

Caso a ocorrência de compilação aponte diretamente para a linha do código-fonte, onde está escrita a instrução EndSql, verifique se existe algum espaço em branco ou tabulação, a partir do início da linha, antes da instrução EndSql. A versão atual desse ambiente não suporta esta declaração, exigindo que a instrução EndSql seja alinhada à esquerda do fonte, sem espaços ou tabulações.

## **Erros de Execução**

### **'Query Argument Error : Alias [XXX] already in use.'**

Caso a instrução BeginSQL especifique um alias que já esteja aberto (em uso), a aplicação é abortada com a ocorrência de erro fatal acima, informando em XXX o alias utilizado.

### **'Query Argument Error : Invalid Value Type [X]'**

Caso alguma expressão informada na Query, através da tag %exp: ... %, retorne um valor de tipo diferente de 'C' Carácter, 'D' Data, 'N' Numérico, ou 'L' Lógico, a aplicação é abortada com a ocorrência de erro acima, onde o tipo do argumento inesperado é mostrado em [X].

### **'Type Mismatch on +'**

Esta ocorrência, se reproduzida, informará na pilha de chamadas o número da linha do código-fonte correspondente à instrução EndSQL. Ocorre caso alguma função intermediária do engine do Embedded SQL, excluindo-se as funções especificadas na query com a sintaxe %exp: ... %, retornar um conteúdo não-caractere a ser acrescentado na Query. Esta ocorrência é de mais difícil localização, sendo útil nestes casos a análise do arquivo temporário gerado pelo Protheus IDE, no momento da compilação.

### **Help NOFUNCW - Função \_\_EXECSQL**

Caso um fonte com o Embedded SQL seja executado em um repositório que não tenha sido atualizado, OU que não seja um Re却itório para o ambiente TOPConnect ( RPODB=TOP ), a aplicação exibirá a ocorrência acima, indicando que a função interna de execução da Query não está presente no ambiente. Verifique se a lib está atualizada, e se o RPO em uso é de um ambiente TOPConnect.

## **Características operacionais - depuração**

---

Dada a montagem da Query, não é possível depurar o bloco do código-fonte compreendido entre as instruções BeginSql e EndSql, não sendo considerados pontos de parada de depuração (*BreakPoints*), caso colocados neste intervalo do código. A colocação de pontos de parada deve ser realizada antes ou depois deste bloco.

### **Função auxiliar - GETLastQuery()**

---

Após a abertura do cursor, no alias especificado, a função GetLastQuery() retorna um array, com 5 elementos, onde estão disponíveis as seguintes informações sobre a Query executada :

- [1] cAlias - Alias usado para abrir o Cursor.
- [2] cQuery - Query executada.
- [3] aCampos - Array de campos com critério de conversão especificados.
- [4] lNoParser - Caso .T., não foi utilizada ChangeQuery() na String original.
- [5] nTimeSpend - Tempo, em segundos, utilizado para a abertura do Cursor.

#### **Exemplo mais completo: Código ADVPL**

```
BeginSql alias 'E2TEMP'
    column E2_EMISSAO as Date, E2_VALOR as Numeric(tam_cp,2)
    column QEK_SKLDOC As Logical

    %noparser%

    SELECT SE2.E2_PREFIXO,SE2.E2_NUM, SE2.E2_FORNECE, SE2.E2_LOJA,SE2.E2_VALOR,
    SE2.D_E_L_E_T_ DEL1, QEK.D_E_L_E_T_ DEL2 , QEK.QEK_SKLDOC, SE2.R_E_C_N_O_
    SE2RECNO
    FROM %table:SE2% SE2,%table:qeK% QEK
    WHERE SE2.E2_FILIAL= %xfilial:SE2% AND
        qek.%notDel% and
        SE2.E2_PREFIXO<> %exp:cPrefixo% AND
        SE2.E2_NUM<> %exp:(cAlias)->M0_CODIGO% AND
        SE2.E2_NUM<>45 AND
        SE2.E2_FORNECE=%exp:Space(Len(SE2->E2_FORNECE))% AND
        SE2.E2_EMISSAO<>%exp: MV_PAR06% AND
        SE2.E2_LOJA<>%exp: MV_PAR05% AND
        SE2.E2_VALOR<>%exp: MV_PAR04% AND
        qek.QEK_SKLDOC<>%exp: MV_PAR03% And
        SE2.%notDel%
    ORDER BY %Order:SE2,1%
EndSql
```

**Exemplo mais completo: Fonte gerado pelo pré-compilador (PPO)**

```
__execSql(
    'E2TEMP',
    ' SELECT SE2.E2_PREFIXO,SE2.E2_NUM, SE2.E2_FORNECE, SE2.E2_LOJA,SE2.E2_VALOR,
    SE2.D_E_L_E_T_ DEL1, QEK.D_E_L_E_T_ DEL2 , QEK.QEK_SKLDOC, SE2.R_E_C_N_O_
    SE2RECNO FROM  '+RetSqlName('SE2')+ ' SE2, '+RetSqlName('QEK')+' QEK WHERE
    SE2.E2_FILIAL=  '' +xFilial('SE2')+'' AND qek.D_E_L_E_T_=  '' AND
    SE2.E2_PREFIXO<>  '+__SQLGetValue(CPREFIXO)+' AND SE2.E2_NUM <> '+
    __SQLGetValue((CALIAS)->M0_CODIGO)+' AND SE2.E2_NUM<>45 AND SE2.E2_FORNECE= '+
    __SQLGetValue(SPACE(LEN(SE2->E2_FORNECE)))+' AND SE2.E2_EMISSAO<> '+
    __SQLGetValue(MV_PAR06)+' AND SE2.E2_LOJA<> '+__SQLGetValue(MV_PAR05) +' AND
    SE2.E2_VALOR<> '+__SQLGetValue(MV_PAR04)+' AND qek.QEK_SKLDOC<> '+
    __SQLGetValue(MV_PAR03)+' And SE2.D_E_L_E_T_=  '' ORDER BY  '+
    SqlOrder(SE2->(IndexKey(1))), {{'E2_EMISSAO','D',8,0}, {'E2_VALOR','N',tam_cp,2},
    {'QEK_SKLDOC','L',1,0}},.T.)
```



*Anotações*

---

---

---

---

## **9. Funcionalidade MsExecAuto**

### **Sobre a MsExecAuto e Rotinas Automáticas**

A funcionalidade MsExecAuto, ou também conhecida como Rotina Automática, permite a execução de rotinas do ambiente ERP Protheus por funções específicas, o que confere ao desenvolvedor a possibilidade de efetuar tratamentos específicos antes da execução da rotina padrão, e mais importante, não perder nenhuma funcionalidade que a rotina padrão oferece.

Avaliando esta funcionalidade apenas pelo parágrafo acima, tem-se a impressão de ser um recurso simples e até mesmo desnecessário, pois um desenvolvedor experiente poderia reproduzir todas as validações e tratamentos existentes em sua rotina, então porque ter de tratar a rotina padrão?

Para responder a esta pergunta devemos fazer as seguintes considerações:

**A aplicação ERP está em constante evolução:**

No momento de desenvolvimento da rotina específica era necessário apenas o tratamento de um conjunto isolado de informações, mas com o aprimoramento do ERP como um todo, agora são necessários outros conjuntos de dados, os quais não foram tratados pelo desenvolvedor naquele momento, mas que estão contemplados na rotina padrão do sistema.

Como o desenvolvedor optou por realizar todo o tratamento de forma específica, em uma atualização do ERP este desenvolvimento precisará ser revisto ou até mesmo refeito, o que implicará em custo para o cliente ou para o desenvolvedor.

Se o desenvolvedor tivesse optado por utilizar a rotina padrão encapsulada em seu desenvolvimento ele não teria problemas em adaptar sua rotina as novas necessidades do sistema, pois a rotina padrão já se preocupada com a compatibilidade entre as versões, possibilitando que uma simples manutenção atualize toda a rotina específica, isto se esta manutenção for necessária, pois a rotina padrão poderá tratar as novas informações com "conteúdos padrões" dependendo da situação em que estas não forem informadas.

**A aplicação ERP pode ser personalizada através de pontos de entrada e do dicionário de dados:**

A partir do momento que uma aplicação padrão é desenvolvida e disponibilizada a mesma pode sofrer diversas personalizações em campo, as quais podem ser implementadas em paralelo ou após o desenvolvimento da rotina específica disponibilizada pelo desenvolvedor.

Se esta situação não for prevista, a inclusão de informações utilizando a rotina padrão do sistema poderá sofrer diversas validações ou até mesmo efetuar a gravação de informações adicionais, as quais não estão visíveis na rotina específica, gerando graves inconsistências na base de dados.

Avaliando estas considerações percebesse o fator de sucesso que um desenvolvimento específico pode ter ao optar por refazer todos os tratamentos do sistema em detrimento de utilizar a funcionalidade MsExecAuto, ou seja, qualquer modificação no ambiente, independente do quanto simples pode tornar a rotina específica inadequada.

## **Quando utilizar a funcionalidade MsExecAuto ?**

---

Tendo em vista este cenário fica a pergunta: "Quando utilizar a funcionalidade MsExecAuto ?"

A resposta pode ser óbvia e até conservadora, mas é "Sempre que a funcionalidade estiver disponível naquele processo".

Em resumo, qualquer funcionalidade de interface, seja ela de carga ou contínua, qualquer funcionalidade de atualização, ou seja, qualquer manutenção na base de dados do sistema, sempre que possível, deve utilizar a funcionalidade MsExecAuto.

## **Processos da aplicação ERP com tratamentos para execução por MsExecAuto**

---

Pode-se dizer que as principais rotinas de atualização do ERP Protheus atualmente possuem o tratamento necessário para sua execução por MsExecAuto, e com base nas necessidades de desenvolvimento internas da TOTVS e nas melhorias solicitadas por clientes e analistas de implantação as demais rotinas necessárias são atualizadas para contemplar este tratamento.

O quadro abaixo ilustra um dos últimos cenários de tratamento de MsExecAuto pelas rotinas da aplicação ERP Protheus.

<b>Rotina</b>	<b>Parâmetros</b>
<b>ATFA010</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>ATFA030</b>	(xAutoCab, nOpc)
<b>ATFA035</b>	(xAutoCab, nOpc)
<b>CFGX016</b>	(xAutoCab, xAutoItens)
<b>CTBA015</b>	(nOpcAuto, aAuto)
<b>CTBA016</b>	(nOpcAuto, aAuto)
<b>CTBA020</b>	(aRotAuto, nOpcAuto, aRotItem)
<b>CTBA102</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>CTBA270</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>FATA140</b>	(nOpcAuto, aAuto)
<b>FATA220</b>	(nOpcAuto, aAutoCab)
<b>FATA300</b>	(nOpcAuto, xAutoCab, xAutoAD2, xAutoAD3, xAutoAD4, xAutoAD9)
<b>FATA310</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>FATA320</b>	(xAutoVend, xAutoTask, xAutoCalend)
<b>FINA010</b>	(aRotAuto, nOpcAuto)
<b>FINA040</b>	(aRotAuto, nOpcAuto)
<b>FINA050</b>	(aRotAuto, nOpcion, nOpcAuto)
<b>FINA070</b>	(xAutoCab, nOpc)
<b>FINA080</b>	(xAutoCab, nOpc)
<b>FINA390</b>	(nPosArotina, xAutoCab, nOpcAuto)
<b>MATA040</b>	(aAuto, nOpcAuto)
<b>MATA080</b>	(aRotauto, nOpcAuto)
<b>MATA103</b>	(xAutoCab, xAutoItens, nOpcAuto, lWhenGet, xAutoImp)
<b>MATA105</b>	(xReservCab, xReservItens, nOpcAuto)
<b>MATA110</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA116A</b>	(xAutoCab, xAutoItens, lInclui)
<b>MATA120</b>	(nFuncao, xAutoCab, xAutoItens, nOpcAuto, )
<b>MATA125</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA140</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA145</b>	(xAutoCab, xAutoIt, xAutoIt2, nOpcAuto)
<b>MATA150</b>	(xAutoCab, xAutoItens, nOpcAuto)

<b>Rotina</b>	<b>Parâmetros</b>
<b>MATA175</b>	(xRotAuto, xOpcAuto)
<b>MATA185</b>	(xAutoSCP, xAutoSD3, nOpcAuto, xPerg)
<b>MATA200</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA230</b>	(aAuto, nOpcAuto)
<b>MATA241</b>	(xAutoCab, xAutoItens, nCallOpcx)
<b>MATA242</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA261</b>	(aAutoItens, nOpcAuto)
<b>MATA265</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA266</b>	(aAutoItens)
<b>MATA267</b>	(aAutoItens)
<b>MATA360</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA410</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA415</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA416</b>	(xAutoCab, xAutoItens)
<b>MATA490</b>	(xAuto, nOpcAuto)
<b>MATA685</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>MATA700</b>	(xAuto, nOpcAuto)
<b>MATA920</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>TMKA061</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>TMKA062</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>TMKA271</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>TMSA040</b>	(xAutoCab, xAutoItens, nOpcAuto)
<b>TMSA050</b>	(xAutoCab, xAutoItens, xItensPesM3, xItensEnder, nOpcAuto)
<b>TMSA170</b>	(xAutoCab, nOpcAuto)
<b>TMSA360</b>	(xAutoCab, xAutoItens, xAutoNFAva, nOpcAuto)
<b>TMSA430</b>	(cAlias, xAutoCab, xAutoVei, xAutoMot, nOpcAuto)
<b>TMSA440</b>	(xAutoCab, xAutoItens, nOpcAuto)

### **Quando não utilizar a funcionalidade MsExecAuto**

Apesar de todas as orientações dos tópicos anteriores existem situações nas quais não é recomendado utilizar a funcionalidade MsExecAuto, no caso, em situações para as quais existem rotinas padrões da aplicação ERP para processamentos de integrações e atualizações que possuem tratamento para execução direta ou execução via Job.

Para estes casos a rotina já está pré-disposta a execução direta, não havendo a necessidade de tratamento via MsExecAuto, ou o tratamento realizado pela rotina é o mais adequado para aquele volume de dados.

O exemplo clássico desta situação é para importação de lançamentos ou saldos contábeis. A aplicação ERP possui uma rotina de integração denominada "Contabilização TXT" (CTBA500) mas a rotina de lançamentos contábeis (CTBA102) pode ser executada através da MsExecAuto.

Para importações de saldos e lançamentos deve-se optar pela utilização da rotina de "Contabilização TXT", devido aos tratamentos que a mesma permite através das configurações da própria rotina em conjunto com as parametrizações do módulo. O uso da CTBA102 – Lançamentos Contábeis via MsExecAuto é recomendado para rotinas de manutenção ou funcionalidades específicas / melhorias, com por exemplo o estorno de lançamentos contábeis sem a necessidade de avaliar regras de contabilização.

## **9.1. Desenvolvendo aplicações com MsExecAuto**

### **Estrutura de uma rotina com execução de MsExecAuto**

- Definição das variáveis de controle da MsExecAuto
- Montagem dos arrays de parâmetros
- Definição dos parâmetros específicos da rotina que será executada
- Controle de transação
- Tratamento do sucesso ou não da operação

### **Variáveis de controle**

#### **Private IMsErroAuto**

Indica se houve erro não fatal durante a execução.

#### **Private IMsHelpAuto**

Habilita a captura das mensagens de erro.

#### **Private IAutoErrNoFile**

Desabilita a geração do arquivo de log padrão do sistema.

### **Montagem dos arrays de parâmetros**

#### **Tipos de arrays de parâmetros:**

Basicamente uma MsExecAuto possui dois tipos de arrays de parâmetros: Array de Dados de Cabeçalho e Array de Dados de Itens.

Os exemplos a seguir visam ilustrar a utilização de cada uma dos tipos de arrays:

- Cadastro de clientes (MATA030): Esta rotina atualiza apenas os dados da tabela SA1 – Cadastro de Clientes, portanto será necessário apenas um array de dados contendo as informações que deverão ser gravadas na tabela SA1.
- Documentos de entrada (MATA103): Esta rotina atualiza basicamente os dados das tabelas SF1 – Cabeçalho do Documento de Entrada e SD1 – Itens do Documento de entrada, portanto serão necessários dois arrays de dados contendo as informações do cabeçalho do documento de entrada e dos itens do documento de entrada, sendo que este último terá sua estrutura de conteúdo repetida para cada item do documento.
- Aviso de Recebimento de Carga (MATA145): Esta rotina atualiza basicamente os dados das tabelas DB1 – Aviso de Recebimento de Carga, DB2 – Cabeçalho do documento recebido e DB3 – Itens do documento recebido, portanto são necessários três array de dados contendo as respectivas informações para as tabelas DB1, DB2 e DB3, sendo que as duas últimas terão suas estruturas de conteúdo repetidas para cada documento recebido.

**Estrutura dos arrays de parâmetros:**

**Array de dados do cabeçalho (xAutoCab):**

O array contendo os dados do cabeçalho é composto por: Campo, Conteúdo e Validação.

Cada item deste array é um dos campos da tabela de cabeçalho das informações que serão processadas pela MsExecAuto. Com base nos exemplos anteriores, a MsExecAuto da MATA030 possui apenas um array de parâmetros no formato xAutoCab, sendo que a MATA103 utilizará o xAutoCab para a tabela SF1 e a MATA145 para a tabela DB1.

**Sintaxe:**

```
AADD(xAutoCab, {"Campo", xConteúdo, "Validação"})
```

**Exemplo:**

```
AADD(xAutoCab, {"A1_FILIAL"      , xFilial("SA1")   , Nil})
AADD(xAutoCab, {"A1_COD"        , "000001"       , Nil})
AADD(xAutoCab, {"A1_LOJA"       , "01"           , Nil})
AADD(xAutoCab, {"A1_NOME"       , "TESTE-000001" , Nil})

MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)
```

**Array de dados dos itens (xAutoItens):**

O array contendo os dados dos itens também é composto por: Campo, Conteúdo e Validação; sendo que a diferença está no fato de que para cada item a ser tratado pela MsExecAuto deverá ser repetida e preenchida a estrutura do array com as informações do mesmo.

Com base nos exemplos anteriores, a MsExecAuto da MATA030 não possui um array de parâmetros no formato xAutoItens, já a MATA103 utilizará o xAutoItens para a tabela SD1 e a MATA145 utilizará dois xAutoItens, sendo o primeiro para a tabela DB2 e o segundo para a tabela DB3.

**Sintaxe:**

```
<Estrutura de controle e repetição>

xAutoItem := {}

AADD(xAutoItem, {"Campo", xConteúdo, "Validação"})

AADD(xAutoItens, xAutoItem)

</Estrutura>
```

### **Exemplo:**

```
AADD(xAutoCab, {"F1_FILIAL" , xFilial("SF1") , Nil})
AADD(xAutoCab, {"F1_DOC"    , "000001"      , Nil})
AADD(xAutoCab, {"F1_SERIE"  , "1"           , Nil})
AADD(xAutoCab, {"F1_FORNECE", "000001"      , Nil})
AADD(xAutoCab, {"F1_LOJA"   , "01"          , Nil})

For nX := 1 to 2

    xAutoItem := {}

    AADD(xAutoItem, {"D1_FILIAL" , xFilial("SD1") , Nil})
    AADD(xAutoItem, {"D1_DOC"    , "000001"      , Nil})
    AADD(xAutoItem, {"D1_SERIE"  , "1"           , Nil})
    AADD(xAutoItem, {"D1_FORNECE", "000001"      , Nil})
    AADD(xAutoItem, {"D1_LOJA"   , "01"          , Nil})
    AADD(xAutoItem, {"D1_ITEM"   , STRZERO(nx,04) , Nil})
    AADD(xAutoItem, {"D1_COD"    , STRZERO(nx,15) , Nil})
    AADD(xAutoItem, {"D1_QUANT"  , 100*nX       , Nil})
    AADD(xAutoItem, {"D1_VUNIT"  , 5*nX         , Nil})
    AADD(xAutoItem, {"D1_TOTAL"  , (100*nX)*(5*nX), Nil})
    AADD(xAutoItem, {"D1_TES"    , "001"         , Nil})

    AADD(xAutoItens, xAutoItem)

Next nX

MsExecAuto({|x,y,z| MATA103(x,y,z)}, xAutoCab, xAutoItens, 3)
```

### **Método de atribuição de conteúdo Direto:**

O método direto consiste em atribuir diretamente na montagem do array de parâmetros o conteúdo de cada campo que será tratado pela MsExecAuto.

Este método é o mais comum de ser utilizado dado sua praticidade, e pela possibilidade do desenvolvedor especificar somente os campos obrigatórios para a tabela e aqueles que conterão as informações geradas por sua aplicação.

### **Exemplo:**

```
AADD(xAutoCab, {"A1_FILIAL" , xFilial("SA1") , Nil})
AADD(xAutoCab, {"A1_COD"    , "000001"      , Nil})
AADD(xAutoCab, {"A1_LOJA"   , "01"          , Nil})
AADD(xAutoCab, {"A1_NOME"   , "TESTE-000001" , Nil})

MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)
```

**Método de atribuição de conteúdo pela estrutura:**

O método de atribuição pela estrutura é recomendável em rotinas que efetuam a importação das informações através da leitura de arquivos de dados, sejam eles CODEBASE ou via banco de dados.

Este método consiste em comparar a estrutura do arquivo de origem (a ser importado) contra a do arquivo de destino (tratado pela MsExecAuto). Desta forma os campos que forem coincidentes entre o arquivo de origem e o arquivo de destino serão preenchidos com as informações do arquivo importado, caso contrário os mesmos poderão ser desconsiderados ou serem preenchidos com um conteúdo padrão.

**Exemplo:**

```
// Captura a estrutura da tabela padrão do sistema
DbSelectArea("SN1")
DbSetOrder(1)
aStruSN1 := SN1->(DbStruct())

// Efetua a abertura do arquivo SN1 que será importado
cArqSN1:= cDirArq+"SN1IMP"
IF File(cArqSN1+cFileExt)
    dbUseArea(.T.,,cArqSN1+cFileExt,"SN1IMP",.F.,.F.)
    IndRegua("SN1IMP",cArqSN1+OrdBagExt(),"N1_FILIAL+N1_CBASE+N1_ITEM",;
    ,, "Selecionando Registros...")
ELSE
    HELP("IATF001",1,"HELP","NO_FILE","ARQUIVO SN1IMP não existe."+
    CRLF+"Verifique caminho informado.",1,0)
    RETURN
ENDIF

// Efetua a leitura do arquivo a ser importado
DbSelectArea("SN1IMP")
DbGotop()
WHILE SN1IMP->(!Eof())

    // Efetua a montagem do xAutoCab com base nas estruturas dos
    // arquivos

    FOR nX := 1 TO LEN(aStruSN1)
        IF SN1IMP->(FieldPos(aStruSN1[nX][1]))>0
            DO CASE
                CASE EMPTY(SN1IMP->&(aStruSN1[nX][1]))
                    AADD(xAutoCab,{aStruSN1[nX][1],;
                    CRIAVAR(aStruSN1[nX][1]), NIL})

                CASE aStruSN1[nX][2] == "C"
                    AADD(xAutoCab,{aStruSN1[nX][1],;
                    SN1IMP->&(aStruSN1[nX][1]),;
                    "",aStruSN1[nX][3]), NIL})

                CASE aStruSN1[nX][2] == "N"
                    AADD(xAutoCab,{aStruSN1[nX][1],;
                    ABS(SN1IMP->&(aStruSN1[nX][1])), NIL})
```

```

        OTHERWISE
            AADD(xAutoCab, {aStruSN1[nX][1],;
                           SN1IMP->&(aStruSN1[nX][1]), NIL})
        ENDCASE
    ELSE
        AADD(xAutoCab, {aStruSN1[nX][1],;
                        CRIAVAR(aStruSN1[nX][1]), NIL})
    ENDIF
NEXT nX

...
MsExecAuto({|x,y,z| ATFA010(x,y,z)}, xAutoCab, xAutoItens, 3)
END

```

### **Definição dos parâmetros específicos da rotina que será executada**

Este é ponto no qual o desenvolvedor inclui os tratamentos necessários a correta execução da rotina, além de seus tratamentos específicos, os quais justificaram o desenvolvimento.

Utilizando o mesmo exemplo anterior da ATFA010 – Cadastro de Ativos podemos citar dois tratamentos específicos necessários para a correta execução da MsExecAuto:

**Chamada das perguntas da ATFA010:**

A execução da função **Pergunte("ATFA010",.F.)** permite que seja definidos os conteúdos desejados para os parâmetros da rotina de cadastro conforme a necessidade da aplicação, como por exemplo permitir a inclusão de uma chapa de bem em branco ou não efetuar a contabilização da inclusão.

**Atualização das variáveis de memória do ALIAS "M->" para o SN1:**

A execução da função **RegToMemory("SN1",.T.)** é necessária para a utilização da função CRIAVAR() no método de atribuição pela estrutura.



*Importante*

Muitas vezes uma estrutura que funciona para um MsExecAuto pode não funcionar para outro do mesmo tipo, devido as características individuais de cada rotina. Podemos citar os seguintes exemplos:

**MATA103** (Documento de Entrada): Para tratamento do título financeiro a ser gerado pela nota fiscal quando o parâmetro de atualização de financeiro da TES está ativado, deve ser incluído o seguinte item no array xAutoCab:

```
{"E2_NATUREZ" , "NAT01" ,NIL})
```

**MATA650** (Abertura de ordem de produção): Para que sejam geradas as ordens de produção intermediárias na abertura de uma ordem de produção principal deve ser incluído o seguinte item no array xAutoCab:

```
{"AUTEXPLODE" , "S" ,NIL})
```

## **Controle de transação**

A utilização do controle de transação permite garantir a integridade das informações gravadas pelo processamento.

Para utilização do controle de transação na aplicação podem ser utilizados os seguintes blocos de controle:

### **Begin Transaction ... DisarmTransaction() ... End Transaction**

Um bloco de código determinado pelos comandos Begin Transaction ... End Transaction terá suas informações atualizadas somente se antes do execução do comando End Transaction não for executada a função DisarmTransaction().

Desta forma pode-se implementar um controle de transação por item processado, sendo executado o DisarmTransaction() para aquele elemento em que houve algum problema de processamento.

Seguindo este mesmo raciocínio, caso ocorra um erro fatal na aplicação, somente o item que está em processamento será desconsiderado, pois a aplicação ERP efetua um DisarmTransaction() automaticamente, fechamento as transações pendentes e restaurando as situações anteriores, mas apenas para aqueles processamento protegidos pelo bloco de controle de transação.



Importante

O comando END TRANSACTION não pode ter sua interpretação vinculada a uma condição. Nestes casos ocorrerá um erro de compilação indicando que o bloco aberto pelo comando BEGIN TRANSACTION não foi corretamente fechado.

### **Exemplo:**

```
AADD(xAutoCab, {"A1_FILIAL" , xFilial("SA1") , Nil})
AADD(xAutoCab, {"A1_COD"      , "000001"       , Nil})
AADD(xAutoCab, {"A1_LOJA"    , "01"           , Nil})
AADD(xAutoCab, {"A1_NOME"    , "TESTE-000001" , Nil})

BEGIN TRANSACTION

lMsErroAuto := .F.
MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)

IF lMsErroAuto
    DisarmTransaction()
ENDIF

END TRANSACTION
```

**BeginTran() ... DisarmTransaction() ... EndTran()**

As funções BeginTran() e EndTran() visam permitir que o término da transação seja condicional, eliminando a limitação da estrutura BEGIN TRANSACTION ... END TRANSACTION.

**Exemplo:**

```
AADD(xAutoCab, {"A1_FILIAL"      , xFilial("SA1")   , Nil})
AADD(xAutoCab, {"A1_COD"        , "000001"       , Nil})
AADD(xAutoCab, {"A1_LOJA"       , "01"           , Nil})
AADD(xAutoCab, {"A1_NOME"       , "TESTE-000001" , Nil})

BeginTran()

lMsErroAuto := .F.
MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)

IF lMsErroAuto
    DisarmTransaction()
ELSE
    EndTran()
ENDIF

MsUnlockAll()
```



*Importante*

Neste modelo de controle de transação é recomendável a utilização da função MsUnlockAll() para destravar todos os registros que estejam eventualmente travados.



*Anotações*

---

---

---

---

## Tratamento de mensagens de erro

Com a utilização da funcionalidade MsExecAuto a aplicação ERP disponibiliza diversos métodos para visualização e tratamento das mensagens de erro ocorridas durante a execução da rotina, sendo estes:

- Visualização do evento de erro;
- Gravação do evento de erro em arquivo texto;
- Personalização da gravação do evento de erro.

### Visualização do evento de erro

Para visualização em tela do evento de erro ocorrido durante o processamento da rotina via MsExecAuto deve-se utilizar a função MostraErro(), conforme o exemplo:

```
Private lMsHelpAuto      := .T.  
Private lAutoErrNoFile  := .F.  
  
AADD(xAutoCab, {"A1_FILIAL"      , xFilial("SA1")    , Nil})  
AADD(xAutoCab, {"A1_COD"        , "000001"       , Nil})  
AADD(xAutoCab, {"A1_LOJA"       , "01"           , Nil})  
AADD(xAutoCab, {"A1_NOME"       , "TESTE-000001" , Nil})  
  
BEGIN TRANSACTION  
  
lMsErroAuto := .F.  
MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)  
  
IF lMsErroAuto  
    MostraErro()  
    DisarmTransaction()  
ENDIF  
  
END TRANSACTION
```



Importante

O conteúdo das variáveis PRIVATE de controle da MsExecAuto deve ser configurado conforme abaixo:

```
Private IMsHelpAuto      := .T.  
Private IAutoErrNoFile  := .F.
```

### Gravação do evento de erro em arquivo texto

Para gravação em arquivo no formato texto do evento de erro ocorrido durante o processamento da rotina via MsExecAuto deve-se utilizar a função MostraErro(), conforme o exemplo:

```
Private lMsHelpAuto      := .T.
Private lAutoErrNoFile   := .F.

AADD(xAutoCab, {"A1_FILIAL"      , xFilial("SA1")      , Nil})
AADD(xAutoCab, {"A1_COD"        , "000001"          , Nil})
AADD(xAutoCab, {"A1_LOJA"       , "01"              , Nil})
AADD(xAutoCab, {"A1_NOME"       , "TESTE-000001"    , Nil})

BEGIN TRANSACTION

lMsErroAuto := .F.
MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)

IF lMsErroAuto
    MostraErro("\system\")
    DisarmTransaction()
ENDIF

END TRANSACTION
```

A função MostraErro() possui o parâmetro cPath o qual pode ser informado de duas formas:

- Apenas o diretório: Se for informado apenas o diretório será gerado um arquivo com denominação no formato SCxxxxxx.log, aonde xxxxxx será um número sequencial gerado internamente pelo sistema.
- Informando o diretório e o nome do arquivo: A função irá respeitar o nome do arquivo informado, sobrecrendo o conteúdo anterior.



Dica

Caso seja necessário gerar um único arquivo texto contendo todos os eventos de erro gerados pelos sucessivos processamentos da MsExecAuto, a função MostraErro(cPath) deverá ser chamada apenas uma vez, ao término do processamento da rotina.

Isto ocorre pois a função MostraErro() limpa o cache de eventos de erros controlado pela MsExecAuto.



Dica

## Personalização da gravação do evento de erro

Para processamentos mais volumosos, a geração de diversos arquivos de textos ou até mesmo a geração de um único arquivo texto contendo todos os eventos de erro pode dificultar a análise e correção dos problemas encontrados durante o processamento.

Desta forma é possível personalizar a gravação do evento de erro, de forma que o mesmo seja gerado em um arquivo no formato .DBF, permitindo o vínculo do registro processado com a mensagem de erro gerada.

```
Private lMSHelpAuto      := .F.
Private lAutoErrNoFile   := .T.

AADD(xAutoCab, {"A1_COD"      , "000001"      , Nil})
AADD(xAutoCab, {"A1_LOJA"     , "01"          , Nil})
AADD(xAutoCab, {"A1_NOME"     , "TESTE-000001" , Nil})

// Função específica que cria o arquivo no formato DBF que conterá as
// mensagens de erro.
XDBFLOG()

BEGIN TRANSACTION

lMsErroAuto := .F.
MsExecAuto({|x,y| MATA030(x,y)}, xAutoCab, 3)

IF lMsErroAuto
    // Função que retorna o evento de erro na forma de um array
    aAutoErro := GETAUTOGRLOG()

    // Função específica que converte o array aAutoErro em texto
    // contínuo, com a quantidade de caracteres desejada por linha

    // Função específica que efetua a gravação do evento de erro no
    // arquivo previamente criado.
    XGRVLOG(XCONVERRLOG(aAutoErro))

    DisarmTransaction()
ENDIF

END TRANSACTION
```



Importante

As funções específicas XDBFLOG(), XCONVERRLOG() e XGRVLOG() serão detalhadas no exemplo completo de utilização da função MsExecAuto().

## **Aplicação completa de importação utilizando MsExecAuto: Carga de imobilizado**

```
#INCLUDE "PROTHEUS.CH"

/*
+-----+
| Função | IATF001           | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição | IMPORTACAO DO CADASTRO DE ATIVO IMOBILIZADO |      |
+-----+
| Uso       | Curso ADVPL |      |
+-----+
 */

USER FUNCTION IATF001()

LOCAL oProcess

PRIVATE oMainWnd

//Perguntas para parametrizacao da rotina (PARAMBOX)
//MVParBox01 - Tipo de informação (.DBF/.TXT)
//MVParBox02 - LayOut do arquivo (Padrão / Especifico)
//MVParBox03 - Arquivo de dados (Consulta)
//MVParBox04 - Arquivo de layout (Consulta)
//MVParBox05 - Operacao (Incluir, Alterar, Excluir)
//MVParBox06 - Diretorio de Log (Consulta)
//MVParBox07 - Importa CIAP (SIM/NAO)

PRIVATE aTiposFile := { ".DBF", ".DTC", ".TXT" }
PRIVATE MvParBox01 := 0
PRIVATE MvParBox02 := 0
PRIVATE MvParBox03 := ""
PRIVATE MvParBox04 := ""
PRIVATE MvParBox05 := 0
PRIVATE MvParBox06 := ""
PRIVATE MVParBox07 := 0

PRIVATE _cDirectory := ""

RpcSetType(3)
RpcSetEnv("99", "01", "", {"CT1", "SF9", "SN1", "SN2", "SN3", "SN4", "SN5", "SNG", "SM2"})

SET DATE FORMAT "dd/mm/yyyy"

oMainWnd := TWindow():New( 000, 000, 001, 001, "Importação: Fichas do
imobilizado", "", .T.)
oMainWnd:bInit := { || IIF(IC001PARBX(), (oProcess:= MsNewProcess():New({ |lEnd|
PROCATIVO(.F.,oProcess)}), oProcess:Activate(), oMainWnd:End()), oMainWnd:End()) }
oMainWnd:Activate("ICONIZED")

RpcClearEnv()
RETURN
```

**Continuação:**

```
/*
+-----+
| Função | PROCATIVO      | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição | PROCESSAMENTO DA IMPORTACAO DO CADASTRO DE IMOBILIZADO |      |
+-----+
| Uso      | Curso ADVPL          |      |
+-----+
/*

STATIC FUNCTION PROCATIVO(lEnd,oProcess)

Local aArea           := GetArea()
Local aDadosSN1       := {}
Local aDadosSN3       := {}
Local aDadosSF9        := {}
Local aDadostN1       := {}
Local aDadostN2       := {}
Local aDadostN3       := {}
Local aItemSN3        := {}
Local cArqSN1         := ""
Local cArqSN2         := ""
Local cArqSN3         := ""
Local cArqSA2         := ""
Local aStruSN1        := {}
Local aStruSN2        := {}
Local aStruSN3        := {}
Local aStruSF9        := {}
Local nX               := 0
Local nTotRegs        := 0
Local nProcRegs        := 0
Local nPosDados        := 0
Local cCodFor          := ""
Local cLojaFor          := ""
Local cDescFor          := ""
Local cFilAtu          := cFilAnt
Local cDirArq          := Alltrim(MvParBox03)
Local cHistorico        := ""
Local nX               := 0
Local cFileExt          := aTiposFile[MvParBox01]
Local lImpSF9          := MvParBox07==1
Local lImpSNG          := .F. // Ainda nao implementado
Local cKeyImp          := ""
Local aAutoErro         := {}

// Tratamentos adicionais - arquivos específicos do cliente
Local cArqCTA          := ""
Local cArqSM2          := ""
Local cIndSNG          := ""

Local nIndex           := 0
Local cGrupo            := ""
Local cContab           := ""
Local aEntSNG           := {}
Local cChapa             := "000000"

Private lMsErroAuto    := .F.
Private lMSHelpAuto    := .F.
```

**Continuação:**

```
Private lAutoErrNoFile := .T.

// Garante a abertura dos arquivos padrões do sistema
DbSelectArea("CT1")
DbSetOrder(2) // FILIAL + COD.REDUZIDO

DbSelectArea("SM2")
DbSetOrder(1)

DbSelectArea("SN1")
DbSetOrder(1)
aStruSN1 := SN1->(DbStruct())

DbSelectArea("SN2")
DbSetOrder(1)
aStruSN2 := SN2->(DbStruct())

DbSelectArea("SN3")
DbSetOrder(1)
aStruSN3 := SN3->(DbStruct())

DbSelectArea("SF9")
DbSetOrder(1)
aStruSF9 := SF9->(DbStruct())

// Prepara o arquivo no formato .DBF para gravação dos eventos de erro
XDBFLOG()

DbSelectArea("SNG")
cIndSNG := CRIATRAB(Nil,.F.)
IndRegua("SNG",cIndSNG,"NG_FILIAL+NG_CCONTAB",,,,"Selecionando Registros...")
nIndex := RetIndex()
#IFNDEF TOP
    DbSetIndex(cIndSNG+OrdBagExt())
#ENDIF
DbSetOrder(nIndex)

PERGUNTE("ATF010",.F.)
MV_PAR02 := 1 // Permite repetir chapa definido como sim

IF Select("SN1IMP") > 0
    DbSelectArea("SN1IMP")
    DbCloseArea()
ENDIF
IF Select("SN2IMP") > 0
    DbSelectArea("SN2IMP")
    DbCloseArea()
ENDIF
IF Select("SN3IMP") > 0
    DbSelectArea("SN3IMP")
    DbCloseArea()
ENDIF
IF Select("SF9IMP") > 0
    DbSelectArea("SF9IMP")
    DbCloseArea()
ENDIF
```

**Continuação:**

```
IF Select("CTAIMP") > 0
    DbSelectArea("CTAIMP")
    DbCloseArea()
ENDIF

cArqSN1:= cDirArq+"SN1IMP"
IF File(cArqSN1+cFileExt)
    dbUseArea(.T.,,cArqSN1+cFileExt,"SN1IMP",.F.,.F.)
    IndRegua("SN1IMP",cArqSN1+OrdBagExt(),"N1_FILIAL+N1_CBASE+N1_ITEM",,,;
              "Selecionando Registros...")
ELSE
    HELP("IATF001",1,"HELP","NO_FILE","ARQUIVO SN1IMP não existe."+CRLF+;
          "Verifique caminho informado.",1,0)
    RETURN
ENDIF

cArqSN2:= cDirArq+"SN2IMP"
IF File(cArqSN2+cFileExt )
    dbUseArea(.T.,,cArqSN2+cFileExt,"SN2IMP",.F.,.F.)
    IndRegua("SN2IMP",cArqSN2+OrdBagExt(),;
              "N2_FILIAL+N2_CBASE+N2_ITEM+N2_TIPO+N2_SEQ",,,,"Selecionando Registros...")
ELSE
    HELP("IATF001",1,"HELP","NO_FILE","ARQUIVO SN2IMP não existe."+CRLF+;
          "Verifique caminho informado.",1,0)
    RETURN
ENDIF

cArqSN3:= cDirArq+"SN3IMP"
IF File(cArqSN3+cFileExt)
    dbUseArea(.T.,,cArqSN3+cFileExt,"SN3IMP",.F.,.F.)
    IndRegua("SN3IMP",cArqSN3+OrdBagExt(),"N3_FILIAL+N3_CBASE+N3_ITEM",,,;
              "Selecionando Registros...")
ELSE
    HELP("IATF001",1,"HELP","NO_FILE","ARQUIVO SN3IMP não existe."+CRLF+;
          "Verifique caminho informado.",1,0)
    RETURN
ENDIF

IF lImpSF9
    cArqSF9:= cDirArq+"SF9IMP"
    IF File(cArqSF9+cFileExt )
        dbUseArea(.T.,,cArqSF9+cFileExt,"SF9IMP",.F.,.F.)
        IndRegua("SF9IMP",cArqSF9+OrdBagExt(),"F9_FILIAL+F9_CODIGO",,,;
                  "Selecionando Registros...")
    ELSE
        HELP("IATF001",1,"HELP","NO_FILE","ARQUIVO SF9IMP não existe."+
              CRLF+"Verifique caminho informado.",1,0)
        RETURN
    ENDIF
ENDIF
```

## Continuação:

**Continuação:**

```
While SN1IMP->(!Eof())  
  
    nProcRegs++  
    oProcess:IncRegual ("Processando item: "+CValToChar(nProcRegs)+" / "+;  
                         CValToChar(nTotRegs))  
    oProcess:IncRegua2("Ficha de Ativo: ")  
  
    aDadosSN1 := {}  
    aDadosSN3 := {}  
  
    // Compatibilização para utilização do CRIAVAR()  
    REGTOMEMORY("SN1", .T.)  
  
    // Montagem do array com dados sem tratamento e com as informações da IMP  
  
    FOR nX := 1 TO LEN(aStruSN1)  
        IF SN1IMP->(FieldPos(aStruSN1[nX][1]))>0  
            DO CASE  
                CASE EMPTY(SN1IMP->&(aStruSN1[nX][1]))  
                    AADD(aDadosSN1,{aStruSN1[nX][1],;  
                               CRIAVAR(aStruSN1[nX][1]), NIL})  
                CASE aStruSN1[nX][2] == "C"  
                    AADD(aDadosSN1,{aStruSN1[nX][1],;  
                               SN1IMP->&(aStruSN1[nX][1]),;  
                               "",aStruSN1[nX][3]), NIL})  
                CASE aStruSN1[nX][2] == "N"  
                    AADD(aDadosSN1,{aStruSN1[nX][1],;  
                               ABS(SN1IMP->&(aStruSN1[nX][1])), NIL})  
                OTHERWISE  
                    AADD(aDadosSN1,{aStruSN1[nX][1],;  
                               SN1IMP->&(aStruSN1[nX][1]), NIL})  
            ENDCASE  
        ELSE  
            AADD(aDadosSN1,{aStruSN1[nX][1], CRIAVAR(aStruSN1[nX][1]),;  
                               NIL})  
        ENDIF  
    NEXT nX  
  
    // Alteração das informações do array de acordo com a necessidade  
    // AADD(aDadosTN1, {"CAMPO", CONTEUDO, "VALIDACAO"})  
    aDadosTN1 := {}  
  
    // ALTERACAO DO ARRAY ADADOSSN1 COM AS INFORMACOES TRABALHADAS E  
    // ARMAZENADAS EM ADADOSTN1  
    // AADD(aDadosTN1, {"N1_XXXXXX", ,xConteudo, ,Nil})  
  
    For nX := 1 TO LEN(aDadosTN1)  
        IF (nPosField := aScan(aDadosSN1, {|aDadosSN1| aDadosSN1[1] ==;  
                                         aDadosTN1[nX][1]})) > 0  
            aDadosSN1[nPosField][2] := aDadosTN1[nX][2]  
            aDadosSN1[nPosField][3] := aDadosTN1[nX][3]  
        ENDIF  
    Next nX
```

**Continuação:**

```
IF SN1->(DbSeek(aDadosSN1[1][2]+aDadosSN1[2][2]+aDadosSN1[3][2]))
    ConOut ("Registro já importado: "+;
            aDadosSN1[1][2]+"/"+aDadosSN1[2][2]+"/"+aDadosSN1[3][2])
    ConOut ("Registros importados com sucesso: "+CValToChar(nProcRegs))
    SN1IMP->(dbSkip())
    Loop
ENDIF

SN3IMP->(DbSeek(SN1IMP->(N1_FILIAL+N1_CBASE+N1_ITEM)))
While SN3IMP->(!Eof()) .AND. SN3IMP->(N3_FILIAL+N3_CBASE+N3_ITEM) ==;
    SN1IMP->(N1_FILIAL+N1_CBASE+N1_ITEM)

    aItensSN3 := {}

    //³ Monstagem do array com dados sem tratamento e com as informações

    FOR nX := 1 TO LEN(aStruSN3)
        IF SN3IMP->(FieldPos(aStruSN3[nX][1]))>0
            DO CASE
                CASE EMPTY(SN3IMP->&(aStruSN3[nX][1]))
                    AADD(aItensSN3,{aStruSN3[nX][1],;
                        CRIAVAR(aStruSN3[nX][1]), NIL})
                CASE aStruSN3[nX][2] == "C"
                    AADD(aItensSN3,{aStruSN3[nX][1],;
                        SN3IMP->&(aStruSN3[nX][1]),;
                        aStruSN3[nX][3]), NIL})
                CASE aStruSN3[nX][2] == "N"
                    AADD(aItensSN3,{aStruSN3[nX][1],;
                        ABS(SN3IMP->&(aStruSN3[nX][1])),;
                        ".T."})

                OTHERWISE
                    AADD(aItensSN3,{aStruSN3[nX][1],;
                        SN3IMP->&(aStruSN3[nX][1]), NIL})

                ENDCASE
            ELSEIF aStruSN3[nX][2] == "N"
                AADD(aItensSN3,{aStruSN3[nX][1],;
                    CRIAVAR(aStruSN3[nX][1]), ".T."})
            ELSE
                AADD(aItensSN3,{aStruSN3[nX][1],;
                    CRIAVAR(aStruSN3[nX][1]), NIL})
            ENDIF
        NEXT nX

        // Alteração das informações do array de acordo com a necessidade
        // AADD(aDadosTN3, {"CAMPO"      , CONTEUDO, "VALIDACAO"})
        aDadosTN3 := {}

        // ALTERACAO DO ARRAY aItensSN3 COM AS INFORMACOES TRABALHADAS E
        // ARMAZENADAS EM aDadosTN3

        For nX := 1 TO LEN(aDadosTN3)
            IF (nPosField := aScan(aItensSN3, {|aItensSN3| aItensSN3[1] ==;
                aDadosTN3[nX][1]})) > 0
                aItensSN3[nPosField][2] := aDadosTN3[nX][2]
                aItensSN3[nPosField][3] := aDadosTN3[nX][3]
```

**Continuação:**

```
        ENDIF
    Next nX

    AADD(aDadosSN3,aItensSN3)
    SN3IMP->(DbSkip())
Enddo

IF      Len(aDadosSN1) > 0 .AND. Len(aDadosSN3) > 0
    ConOut("Iniciando MsExecAuto - ATFA010: "+Time())

    Begin Transaction
    cFilAnt := SN1IMP->N1_FILIAL
    lMsErroAuto := .F.
    MSExecAuto({|x,y,z| Atfa010(x,y,z)},aDadosSN1,aDadosSN3,3)
    // Cabeçalho, Itens e Opção

    ConOut("Finalizada MsExecAuto - ATFA010: "+Time())

    IF lMsErroAuto
        aAutoErro := GETAUTOGRLLOG()
        XGRVLOG(cKeyImp,;
                  SN1IMP->(N1_FILIAL+N1_CBASE+N1_ITEM),;
                  SN1IMP->N1_DESCRIC, XCONVERRLOG(aAutoErro))
        DisarmTransaction()
        MostraErro(Alltrim(MVParBox06))
    ELSE
        // TRATAMENTO DA DESCRIÇÃO ESTENDIDA (SN2)
        SN2IMP->(DbSeek(SN1IMP->(N1_FILIAL+N1_CBASE+N1_ITEM)))
        cHistorico := ALLTRIM(SN2IMP->N2_HISTOR1) +
                      ALLTRIM(SN2IMP->N2_HISTOR2)

        For nX := 1 to Len(cHistorico) STEP 40
            RECLKLOCK("SN2",.T.)
            SN2->N2_FILIAL     := SN1->N1_FILIAL
            SN2->N2_CBASE      := SN1->N1_CBASE
            SN2->N2_ITEM       := SN1->N1_ITEM
            SN2->N2_TIPO        := "01"
            SN2->N2_SEQ         := "001"
            SN2->N2_SEQUENC     := STRZERO(nX,2)
            SN2->N2_HISTOR     := SUBSTR(cHistorico,nX,40)
            MSUNLOCK()

            NEXT nX

            // TRATAMENTO DA CARGA DO CIAP
            IF lImpSF9
                IATFCIAP(aStruSF9,aDadosSF9)
            ENDIF

            ConOut("Registros importados com sucesso: "+;
                   CValToChar(nProcRegs))

        ENDIF
    End Transaction
```

**Continuação:**

```
        ENDIF

        SN1IMP->(DbSkip())
Enddo

oProcess:IncRegual("Processamento finalizado")

DbSelectArea("SN1IMP")
DbCloseArea()
IF File(cArqSN1+OrdBagExt())
    FErase(cArqSN1+OrdBagExt())
ENDIF

DbSelectArea("SN2IMP")
DbCloseArea()
IF File(cArqSN2+OrdBagExt())
    FErase(cArqSN2+OrdBagExt())
ENDIF

DbSelectArea("SN3IMP")
DbCloseArea()
IF File(cArqSN3+OrdBagExt())
    FErase(cArqSN3+OrdBagExt())
ENDIF

IF lImpSF9
    DbSelectArea("SF9IMP")
    DbCloseArea()
    IF File(cArqSF9+OrdBagExt())
        FErase(cArqSF9+OrdBagExt())
    ENDIF
ENDIF

IF lImpSNG
    DbSelectArea("SNGIMP")
    DbCloseArea()
    IF File(cIndSNG+OrdBagExt())
        FErase(cIndSNG+OrdBagExt())
    ENDIF
ENDIF

DbSelectArea("CTAIMP")
DbCloseArea()
IF File(cArqCTA+OrdBagExt())
    FErase(cArqCTA+OrdBagExt())
ENDIF

DbSelectArea("SM2IMP")
DbCloseArea()
IF File(cArqSM2+OrdBagExt())
    FErase(cArqSM2+OrdBagExt())
ENDIF
```

**Continuação:**

```
DbSelectArea("LOGIMP")
DbCloseArea()

ConOut("Total de registros importados: "+CValToChar(nProcRegs))
ConOut("Término da importação: "+Time())

cFilAnt := cFilAtu
RestArea(aArea)
RETURN

/*
+-----
| Função      | CT001PARBX      | Autor | Arnaldo R. Junior | Data |      |
+-----
| Descrição   | TELA DE PARAMETROS ESPECIFICOS DA ROTINA CUSTOMIZADA |      |
+-----|
| Uso         | Curso ADVPL          |      |
+-----|
*/
STATIC FUNCTION IC001PARBX()

LOCAL aParamBox    := {}
LOCAL cTitulo       := "Importacao de cadastros"
LOCAL aRet          := {}
LOCAL bOk           := {|| .T.}
LOCAL aButtons     := {}
LOCAL lCentered    := .T.
LOCAL nPosx         :=
LOCAL nPosy         :=
LOCAL cLoad          := ""
LOCAL lCanSave      := .T.
LOCAL lUserSave     := .T.
LOCAL nX             := 0
LOCAL lRet           := .F.

AADD(aParamBox,{2,"Tipo de informação"      ,1 ,aTiposFile ,060,;
               "AllwaysTrue()", .T.})
AADD(aParamBox,{2,"Layout do arquivo "      ,1 ,;
               {"Padrão","Especifico"},060, "AllwaysTrue()", .T.})
AADD(aParamBox,{1,"Diretorio de dados"      ,Space(60)      ,;
               "@!","AllwaysTrue()", "HSSDIR" ,".T.",120,.T.})
AADD(aParamBox,{1,"Arquivo de layout "      ,Space(60)      ,;
               "@!","AllwaysTrue()", "", ".T.",120,.F.})
AADD(aParamBox,{2,"Operacao"                  ,1 ,;
               {"Incluir","Alterar","Excluir"},060, "AllwaysTrue()", .T.})
AADD(aParamBox,{1,"Diretorio de Log"        ,Space(60)      ,;
               "@!","AllwaysTrue()", "HSSDIR" ,".T.",120,.F.})
AADD(aParamBox,{2,"Importa CIAP"            ,1 ,;
               {"Sim","Não"} ,060, "AllwaysTrue()", .T.})

lRet := ParamBox(aParamBox, cTitulo, aRet, bOk, aButtons, lCentered, nPosx,;
nPosy, /*oMainDlg*/ , cLoad, lCanSave, lUserSave)
```

**Continuação:**

```
IF ValType(aRet) == "A" .AND. Len(aRet) == Len(aParamBox)
    For nX := 1 to Len(aParamBox)
        If aParamBox[nX][1] == 1
            & ("MvParBox"+StrZero(nX,2)) := aRet[nX]
        ElseIf aParamBox[nX][1] == 2 .AND. ValType(aRet[nX]) == "C"
            & ("MvParBox"+StrZero(nX,2)) := aScan(aParamBox[nX][4],;
            { |x| Alltrim(x) == aRet[nX] })
        ElseIf aParamBox[nX][1] == 2 .AND. ValType(aRet[nX]) == "N"
            & ("MvParBox"+StrZero(nX,2)) := aRet[nX]
        Endif
    Next nX
ENDIF

RETURN lRet

/*
+-----+
| Função | IATFCIAP | Autor | Arnaldo R. Junior | Data | |
+-----+
| Descrição | IMPORTACAO DO LIVRO FISCAL CIAP | |
+-----+
| Uso | Curso ADVPL | |
+-----+
*/
STATIC FUNCTION IATFCIAP(aStruSF9,aDadosSF9)
Local aDadosCIAP := {}
Local nX := 0
Local nPosSF9 := 0

// Monta array com dados padrões do SF9 de acordo com o SX3
FOR nX := 1 to Len(aStruSF9)
    AADD(aDadosCIAP,{aStruSF9[nX][1],CRIAVAR(aStruSF9[nX][1]),NIL})
NEXT nX

// Atualiza dados do array com as informações presentes no SN1
FOR nX := 1 to Len(aDadosSF9)
    IF (nPosSF9 := aScan(aDadosCIAP,{|aLinhaCIAP| aLinhaCIAP[1] ==
aDadosSF9[nX][1]})) > 0
        aDadosCIAP[nPosSF9][2] := aDadosSF9[nX][2]
    ENDIF
NEXT nX

ConOut("Iniciando MsExecAuto - ATFCIAP: "+Time())
lMsErroAuto := .F.
MSExecAuto({|x,y| U_ATFCIAP(x,y)},aDadosCIAP,3) // Dados e Opção
ConOut("Finalizada MsExecAuto - ATFCIAP: "+Time())

RETURN lMsErroAuto
```

**Continuação:**

```
/*
+-----+
| Função      | XDBFLOG          | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição  | CRIACAO DO ARQUIVO DBF PARA TRATAMENTO DOS EVENTOS DE ERR|
+-----+
| Uso        | Curso ADVPL          |
+-----+
*/
STATIC FUNCTION XDBFLOG()
LOCAL aCampos      := {}
LOCAL cArqLog      := MVParBox06+"LOGIMP"+GetDbExtension()

IF !File(cArqLog)
    AADD(aCampos, {"CKEYIMP", "C", 014, 0})
    AADD(aCampos, {"CKEYREG", "C", 020, 0})
    AADD(aCampos, {"CDESCR", "C", 040, 0})
    AADD(aCampos, {"CSEQMSG", "C", 003, 0})
    AADD(aCampos, {"CMMSGERR", "C", 254, 0})
    dbCreate(cArqLog, aCampos, __LocalDriver)
ENDIF

dbUseArea(.T., __LocalDriver, cArqLog, "LOGIMP", .T., .F.)
RETURN

/*
+-----+
| Função      | XGRVLOG          | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição  | GRAVACAO DA MENSAGEM DE ERRO NO ARQUIVO DBF DE CONTROLE |
+-----+
| Uso        | Curso ADVPL          |
+-----+
*/
STATIC FUNCTION XGRVLOG(cKeyImp, cKeyReg, cDescReg, cMsgErr)

LOCAL cSeqLog := "000"

FOR nX := 1 TO Len(cMsgErr) STEP 254

    cSeqLog := SOMA1(cSeqLog)
    RECLOCK("LOGIMP", .T.)
    LOGIMP->CKEYIMP   := cKeyImp
    LOGIMP->CKEYREG   := cKeyReg
    LOGIMP->CDESCR    := cDescReg
    LOGIMP->CSEQMSG   := cSeqLog
    LOGIMP->CMMSGERR  := SUBSTR(cMsgErr, nX, 254)
    MSUNLOCK()

NEXT nX

RETURN
```

**Continuação:**

```
/*
+-----+
| Função | XCONVERRLOG | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição | CONVERTE O ARRAY AAUTOERRO EM TEXTO CONTINUO. |
+-----+
| Uso | Curso ADVPL |
+-----+
*/
STATIC FUNCTION XCONVERRLOG(aAutoErro)

LOCAL cRet := ""
LOCAL nX := 1

FOR nX := 1 to Len(aAutoErro)
    cRet += aAutoErro[nX]+CHR(13)+CHR(10)
NEXT nX

RETURN cRet
```



Anotações

---

---

---

---

## 10. Recursos de envio de e-mail

Neste tópico serão descritas as funções da linguagem ADVPL que permitem o envio e o recebimento de mensagens através de e-mails.

### Funções para manipulação de e-mails

- CALLPROC**
- MAILSMTPON**
- MAILPOPON**
- MAILSMTPOFF**
- MAILPOPOFF**
- MAILRECEIVE**
- MAILAUTH**
- POPMSGCOUNT**
- MAILSEND**
- MAILGETERR**

### Detalhamento das funções de manipulação de e-mails

#### CALLPROC()

Método do objeto oRpcSrv que permite a utilização de todas as funções de envio e recebimento de e-mail descritas neste tópico.

- Sintaxe:** CallProc(**cFuncao**, **xParam1**, ..., **xParamN**)

- Parâmetros:**

<b>cFuncao</b>	Nome da função ou método do objeto oRpcSrv que será executado.
<b>xParamN</b>	Parâmetros da função ou método que será executado.

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **MAILSMTPON()**

Ativa uma conexão com o serviço de SMTP para a thread ativa.

- Sintaxe:** MailSmtpOn( cServer, cUser, cPass, nTimeOut)

- Sintaxe clássica:**

```
CONNECT SMTP SERVER cServer ACCOUNT cUser PASSWORD cPass TIMEOUT nTimeOut IN
SERVER oRpcSrv RESULT lResult
```

- Parâmetros:**

<b>Cserver</b>	Endereço do servidor para conexão
<b>Cuser</b>	Nome do usuário para autenticação no servidor.
<b>CPass</b>	Senha do usuário para autenticação no servidor.
<b>NTimeOut</b>	Tempo de espera para operação de autenticação.

- Retorno:**

<b>Lógico</b>	Indica se foi realizada com sucesso a conexão com o servidor indicado.
---------------	--

## **MAILPOPON()**

Ativa uma conexão com o serviço de POP para a thread ativa. -> lReturn

- Sintaxe:** MailPopOn( cServer, cUser, cPass, nTimeOut)

- Sintaxe clássica:**

```
CONNECT POP SERVER cServer ACCOUNT cUser PASSWORD cPass TIMEOUT nTimeOut IN
SERVER oRpcSrv RESULT lResult
```

- Parâmetros:**

<b>Cserver</b>	Endereço do servidor para conexão
<b>Cuser</b>	Nome do usuário para autenticação no servidor.
<b>CPass</b>	Senha do usuário para autenticação no servidor.
<b>NTimeOut</b>	Tempo de espera para operação de autenticação.

- Retorno:**

<b>Lógico</b>	Indica se foi realizada com sucesso a conexão com o servidor indicado.
---------------	--

## **MAILSMTPOFF()**

Encerra a conexão com o serviço de SMTP.

- Sintaxe: MailSmtpOff()**

- Sintaxe clássica:**

```
DISCONNECT SMTP SERVER IN SERVER oRpcSrv RESULT lResult
```

- Parâmetros:**

**Nenhum**

.

- Retorno:**

**Lógico**

Indica se a operação foi realizada com sucesso.

## **MAILPOPOFF()**

Encerra a conexão com o serviço de POP

- Sintaxe: MailPopOff()**

- Sintaxe clássica:**

```
DISCONNECT POP SERVER IN SERVER oRpcSrv RESULT lResult
```

- Parâmetros:**

**Nenhum**

.

- Retorno:**

**Lógico**

Indica se a operação foi realizada com sucesso.

## **MAILRECEIVE()**

Efetua o recebimento de um e-mail, salvando-o no local definido.

- Sintaxe:** **MailReceive(nNumber, @cFrom, @cTo, @cCc, @cBcc, @cSubject, @cBody, @aFiles, cPath, lDelete)**

- Sintaxe clássica:**

```
RECEIVE MAIL MESSAGE nNumber FROM cFrom TO      cTo CC cCc BCC cBcc SUBJECT
cSubject BODY cBody ATTACHMENT aFiles SAVE IN cPath DELETE IN SERVER oRpcSrv
RESULT lResult
```

- Parâmetros:**

<b>nNumber</b>	Número da mensagem que deverá ser recebida. Este número é em função da quantidade de mensagens na caixa de e-mails.
<b>cFrom</b>	Variável local do fonte que será atualizada com o remetente da mensagem.
<b>cTo</b>	Variável local do fonte que será atualizada com o destinatário da mensagem.
<b>cCc</b>	Variável local do fonte que será atualizada com a conta copiada na mensagem.
<b>cBcc</b>	Variável local do fonte que será atualizada com a conta copiada em cópia oculta na mensagem.
<b>cSubject</b>	Variável local do fonte que será atualizada com o assunto da mensagem.
<b>cBody</b>	Variável local do fonte que será atualizada com corpo da mensagem.
<b>aFiles</b>	Variável local do fonte que será atualizada os nomes dos anexos da mensagem.
<b>cPath</b>	Diretório no qual serão salvos os anexos da mensagem.
<b>lDelete</b>	Indica se a mensagem deverá ser apagada do servidor de e-mails após sua recepção pela aplicação.

- Retorno:**

<b>Lógico</b>	Indica se a operação de recebimento de mensagens foi realizada com sucesso.
---------------	---



*Anotações*

---



---



---



---



---

## MAILAUTH()

Função que executa a autenticação do usuário no serviço ativo.

- Sintaxe:** MailAuth(cUser, cPassword)

- Parâmetros:**

<b>cUser</b>	Nome do usuário para validação da conexão com o servidor.
<b>cPassword</b>	Senha do usuário para validação da conexão com o servidor.

- Retorno:**

<b>Lógico</b>	Indica se foi realizada a autenticação do usuário com sucesso.
---------------	--



*Importante*

A função MailAuth() deverá ser utilizada obrigatoriamente após a abertura da conexão com o servidor, seja ele de envio ou recebimento de mensagens.



*Dica*

Para validação da conexão sempre efetue a verificação com o usuário contendo o endereço completo do e-mail, e caso necessário somente com o usuário, eliminando o restante do endereço após o caractere "@".

```
IResult := MailAuth(Alltrim(cEmail), Alltrim(cPass))
// Se nao conseguiu fazer a Autenticacao usando o E-mail completo, tenta
// fazer a autenticacao usando apenas o nome de usuario do E-mail.
If !IResult
    nA := At("@",cEmail)
    cUser := If(nA>0,Subs(cEmail,1,nA-1),cEmail)
    IResult := MailAuth(Alltrim(cUser), Alltrim(cPass))
Endif
```



*Anotações*

---

---

---

---

## **POPMSGCOUNT()**

Verifica quantas mensagens existem na caixa de entrada do serviço POP ativo.

- Sintaxe: PopMsgCount(@nMsgCount)**

- Sintaxe clássica:**

```
POP MESSAGE COUNT nMsgCount IN SERVER oRpcSrv RESULT lResult
```

- Parâmetros:**

<b>nMsgCount</b>	Variável local do fonte que será atualizada com a quantidade de mensagens disponíveis para recebimento na caixa de e-mails.
------------------	---

- Retorno:**

<b>Lógico</b>	Indica se foi realizado o acesso a caixa de mensagens.
---------------	--

## **MAILSEND()**

Envia um e-mail utilizando o serviço de SMTP ativo.

- Sintaxe: MailSend(cFrom, aTo, aCc, aBcc, cSubject, cBody, aFiles, lText)**

- Sintaxe clássica:**

```
SEND MAIL FROM cFrom TO aTo,... CC aCc,... BCC aBcc,... SUBJECT cSubject BODY
cBody FORMAT TEXT ATTACHMENT aFiles,... IN SERVER oRpcSrv RESULT lResult
```

- Parâmetros:**

<b>cFrom</b>	Endereço de e-mail do remetente da mensagem.
<b>aTo</b>	Array contendo os endereços de e-mails dos destinatários da mensagem.
<b>aCc</b>	Array contendo os endereços de e-mails dos copiados na mensagem.
<b>aBcc</b>	Array contendo os endereços de e-mails dos copiados de forma oculta na mensagem.
<b>cSubject</b>	Texto de assunto do e-mail.
<b>cBody</b>	Texto da mensagem do e-mail.
<b>aFiles</b>	Array contendo os nomes dos arquivos que serão anexados a mensagem.
<b>lText</b>	Indica se o corpo do e-mail está em formato texto.

- Retorno:**

<b>Lógico</b>	Indica se a operação de envio de mensagens foi realizada com sucesso.
---------------	---

## **MAILGETERR()**

Retorna o erro que ocorreu no envio do e-mail.

**Sintaxe: MailGetErr()**

**Sintaxe clássica:**

```
GET MAIL ERROR cErrorMsg IN SERVER oRpcSrv
```

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Caracter</b>	Mensagem de erro ocorrida no processamento do envio ou recebimento dos e-mails.
-----------------	---

## **Exemplos de utilização das funcionalidades de envio e recebimento de e-mail**

### **Envio de mensagens utilizando sintaxe clássica**

```
#include "protheus.ch"
#include "tbiconn.ch"
#include "AP5MAIL.CH"

/*
+-----+
| Função | SENDMAIL      | Autor | Arnaldo R. Junior | Data |
+-----+
| Descrição | ENVIO DE E-MAIL GENERICO |
+-----+
| Uso       | Curso ADVPL |
+-----+
*/
USER FUNCTION SendMail(_lJob)

Local lResulConn := .T.
Local lResulSend := .T.
Local cError := ""

Local cServer   := AllTrim(GetMV("MV_RELSEERV"))
Local cEmail    := AllTrim(GetMV("MV_RELACNT"))
Local cPass     := AllTrim(GetMV("MV_RELPSW"))
Local lRelauth := GetMv("MV_RELAUTH")

Local cDe       := cEmail
Local cPara    := "arnaldojr@microsiga.com.br"
```

**Continuação:**

```
Local cCc      := ""
Local cAssunto := "Teste de envio de e-mail: Curso ADVPL"
Local cAnexo   := "\SYSTEM\lgr199.bmp"
Local cMsg     := Space(200)

Default _lJob    := .T.

cMsg  := "-----"
cMsg += "CURSO DE ADVPL"
cMsg += "-----"
cMsg += "Você está recebendo um e-mail do curso de ADVPL avançado"

CONNECT SMTP SERVER cServer ACCOUNT cEmail PASSWORD cPass RESULT lResulConn

If !lResulConn
  GET MAIL ERROR cError
  If _lJob
    ConOut(Padc("Falha na conexao "+cError,80))
  Else
    MsgAlert("Falha na conexao "+cError)
  Endif
  Return(.F.)
Endif

// Sintaxe: SEND MAIL FROM cDe TO cPara CC cCc SUBJECT cAssunto BODY cMsg
// ATTACHMENT cAnexo RESULT lResulSend
// Todos os e-mail terão: De, Para, Assunto e Mensagem, porém precisa analisar
// se tem: Com Cópia e/ou Anexo

If lRelauth
  lResult := MailAuth(Alltrim(cEmail), Alltrim(cPass))
  //Se não conseguiu fazer a Autenticacao usando o E-mail completo, tenta fazer
  //a autenticacao usando apenas o nome de usuario do E-mail
  If !lResult
    nA := At("@",cEmail)
    cUser := If(nA>0,Subs(cEmail,1,nA-1),cEmail)
    lResult := MailAuth(Alltrim(cUser), Alltrim(cPass))
  Endif
Endif

If lResult
  If Empty(cCc) .And. Empty(cAnexo)
    SEND MAIL FROM cDe TO cPara SUBJECT cAssunto BODY cMsg RESULT
lResulSend
  Else
    If Empty(cCc) .And. !Empty(cAnexo)
      SEND MAIL FROM cDe TO cPara SUBJECT cAssunto BODY cMsg
      ATTACHMENT cAnexo RESULT lResulSend
    ElseIf !Empty(cCc) .And. !Empty(cAnexo)
      SEND MAIL FROM cDe TO cPara CC cCc SUBJECT cAssunto BODY cMsg
      ATTACHMENT cAnexo RESULT lResulSend
    ElseIf Empty(cCc) .And. Empty(cAnexo)
      SEND MAIL FROM cDe TO cPara CC cCc SUBJECT cAssunto BODY cMsg
      RESULT lResulSend
    Endif
  Endif
Endif
```

**Continuação:**

```
If !lResulSend
    GET MAIL ERROR cError
    If _lJob
        ConOut(Padc("Falha no Envio do e-mail "+cError,80))
    Else
        MsgAlert("Falha no Envio do e-mail " + cError)
    Endif
Endif

Else

    If _lJob
        ConOut(Padc("Falha na autenticação do e-mail: "+cError,80))
    Else
        MsgAlert("Falha na autenticação do e-mail:" + cError)
    Endif

Endif

DISCONNECT SMTP SERVER

IF lResulSend
    If _lJob
        ConOut(Padc("E-mail enviado com sucesso",80))
    Else
        MsgInfo("E-mail enviado com sucesso" + cError)
    Endif
ENDIF

RETURN lResulSend
```



**Anotações**

---

---

---

---

## Envio de mensagens utilizando funções

```
#include "protheus.ch"
#include 'tbiconn.ch'
#include "AP5MAIL.CH"

/*
+-----
| Função      | SENDMAIL      | Autor | Arnaldo R. Junior | Data |      |
+-----
| Descrição   | ENVIO DE E-MAIL GENERICO |      |
+-----|
| Uso         | Curso ADVPL |      |
+-----|
*/

USER FUNCTION FSendMail(_lJob)

Local lResulConn := .T.
Local lResulSend := .T.
Local lResult    := .T.
Local cError     := ""

Local cServer    := AllTrim(GetMV("MV_RELSEERV"))
Local cEmail     := AllTrim(GetMV("MV_RELACNT"))
Local cPass      := AllTrim(GetMV("MV_RELPSW"))
Local lRelauth   := GetMv("MV_RELAUTH")

Local cDe        := cEmail
Local cPara      := "arnaldojr@microsiga.com.br"
Local cCc        := ""
Local cBcc       := ""
Local cAssunto   := "Teste de envio de e-mail: Curso ADVPL"
Local cAnexo     := "\SYSTEM\lgr199.bmp"
Local cMsg       := Space(200)

Default _lJob    := .F.

cMsg  := "-----"
cMsg  += "CURSO DE ADVPL"
cMsg  += "-----"
cMsg  += "Você está recebendo um e-mail do curso de ADVPL avançado"

//CONNECT SMTP SERVER cServer ACCOUNT cEmail PASSWORD cPass RESULT lResulConn
lResulConn := MailSmtOn( cServer, cEmail, cPass)

If !lResulConn
  //GET MAIL ERROR cError
  cError := MailGetErr()
  If _lJob
    ConOut(Padc("Falha na conexao "+cError,80))
  Else
    MsgAlert("Falha na conexao "+cError)
  Endif
  Return(.F.)
Endif
```

**Continuação:**

```
// Sintaxe: SEND MAIL FROM cDe TO cPara CC cCc SUBJECT cAssunto BODY cMsg
ATTACHMENT cAnexo RESULT lResulSend
// Todos os e-mail terão: De, Para, Assunto e Mensagem, porém precisa analisar
se tem: Com Cópia e/ou Anexo

If lRelauth
    lResult := MailAuth(Alltrim(cEmail), Alltrim(cPass))
    //Se não conseguiu fazer a Autenticação usando o E-mail completo, tenta fazer
    a autenticação usando apenas o nome de usuário do E-mail
    If !lResult
        nA := At("@",cEmail)
        cUser := If(nA>0,Subs(cEmail,1,nA-1),cEmail)
        lResult := MailAuth(Alltrim(cUser), Alltrim(cPass))
    Endif

Endif

If lResult
    /*
    If Empty(cCc) .And. Empty(cAnexo)
        SEND MAIL FROM cDe TO cPara SUBJECT cAssunto BODY cMsg RESULT
        lResulSend
    Else
        If Empty(cCc) .And. !Empty(cAnexo)
            SEND MAIL FROM cDe TO cPara SUBJECT cAssunto BODY cMsg
            ATTACHMENT cAnexo RESULT lResulSend
        ElseIf !Empty(cCc) .And. !Empty(cAnexo)
            SEND MAIL FROM cDe TO cPara CC cCc SUBJECT cAssunto BODY cMsg
            ATTACHMENT cAnexo RESULT lResulSend
        ElseIf Empty(cCc) .And. Empty(cAnexo)
            SEND MAIL FROM cDe TO cPara CC cCc SUBJECT cAssunto BODY cMsg
            RESULT lResulSend
        Endif
    Endif
*/
    lResulSend := MailSend(cDe,{cPara},{cCc},{cBcc},cAssunto,cMsg,{cAnexo},.T.)

    If !lResulSend
        //GET MAIL ERROR cError
        cError := MailGetErr()
        If _lJob
            ConOut(Padc("Falha no Envio do e-mail "+cError,80))
        Else
            MsgAlert("Falha no Envio do e-mail " + cError)
        Endif
    Endif

Else
    If _lJob
        ConOut(Padc("Falha na autenticação do e-mail: "+cError,80))
    Else
        MsgAlert("Falha na autenticação do e-mail: " + cError)
    Endif
Endif
```

**Continuação:**

```
//DISCONNECT SMTP SERVER
MailSmtplib()

IF lResulSend
    If _lJob
        ConOut(PadC("E-mail enviado com sucesso", 80))
    Else
        MsgInfo("E-mail enviado com sucesso" + cError)
    Endif
ENDIF

RETURN lResulSend
```



### Anotações

---

---

---

---

---

## Recebimento de mensagens utilizando funções

```
#include "protheus.ch"
#include "tbiconn.ch"
#include "AP5MAIL.CH"

/*
+-----
| Função      | POPMAIL           | Autor | Arnaldo R. Junior | Data | |
+-----
| Descrição   | RECEBIMENTO DE E-MAIL GENERICO | |
+-----
| Uso         | Curso ADVPL | |
+-----
*/

USER FUNCTION FPopMail(_lJob)

Local lResulConn := .T.
Local lResulPop  := .T.
Local lResult    := .T.
Local cError     := ""

Local cServer    := AllTrim(GetMV("MV_RELSEERV"))
Local cEmail     := AllTrim(GetMV("MV_RELACNT"))
Local cPass      := AllTrim(GetMV("MV_RELPSW"))
Local lRelauth   := GetMv("MV_RELAUTH")

Local cDe        := ""
Local cPara      := ""
Local cCc        := ""
Local cBcc       := ""
Local cAssunto   := ""
Local aAnexo     := {}
Local cMsg        := ""
Local cPath      := "\MailBox"
Local nMsgCount := 0
Local nNumber    := 0

Default _lJob     := .F.

lResulConn := MailPopOn( cServer, cEmail, cPass, 1000)

If !lResulConn
  cError := MailGetErr()
  If _lJob
    ConOut(Padc("Falha na conexao "+cError,80))
  Else
    MsgAlert("Falha na conexao "+cError)
  Endif
  Return(.F.)
Endif
```

**Continuação:**

```
/*If lRelauth
    lResult := MailAuth(Alltrim(cEmail), Alltrim(cPass))
    // Se nao conseguiu fazer a Autenticacao usando o E-mail completo, tenta
    // fazer a autenticacao usando apenas o nome de usuario do E-mail
    If !lResult
        nA := At("@",cEmail)
        cUser := If(nA>0,Subs(cEmail,1,nA-1),cEmail)
        lResult := MailAuth(Alltrim(cUser), Alltrim(cPass))
    Endif

Endif*/
If lResult
    PopMsgCount (@nMsgCount)
    For nNumber := 1 to nMsgCount
        lResulPop := MailReceive(nNumber, @cDe, @cPara, @cCc, @cBcc,;
                                @cAssunto, @cMsg, @aAnexo , cPath, .F.)
        If !lResulPop
            cError := MailGetErr()
            If _lJob
                ConOut(Padc("Falha no recebimento do e-mail "+cError,80))
            Else
                MsgAlert("Falha no recebimento do e-mail " + cError)
            Endif
        Else // Salvar mensagem de e-mail
            cMessage := "DE: "+cDe+CRLF
            cMessage += "PARA: "+cPara+CRLF
            cMessage += "CC: "+cCc+CRLF
            cMessage += "BCC: "+cBcc+CRLF
            cMessage += "ASSUNTO: "+cAssunto+CRLF
            cMessage += "MENSAGEM: "+cMsg+CRLF
            MsgInfo(cMessage,"E-mail Recebido")
        Endif
    End-For // Next nNumber
Else
    If _lJob
        ConOut(Padc("Falha na autenticação do e-mail: "+cError,80))
    Else
        MsgAlert("Falha na autenticação do e-mail: " + cError)
    Endif
Endif
MailPopOff()
```

**Continuação:**

```
IF lResulPop
    If _lJob
        ConOut(Padc("E-mails recebidos com sucesso",80))
    Else
        MsgInfo("E-mail recebidos com sucesso" + cError)
    Endif
ENDIF

RETURN lResulPop
```

## **11. Integração básica com MsOffice**

### **11.1. Exportação para 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(cOrigem, cTitulo, aDadosCab, aDadosItens)**

- Parâmetros:**

<b>cOrigem</b>	Conteúdo fixo definido como: "GETDADOS"
<b>cTitulo</b>	Nome para exibição da planilha
<b>aDadosCab</b>	Array contendo os nomes dos campos que serão exibidos na planilha.
<b>aDadosItens</b>	Array contendo as informações dos campos, de acordo com a ordem do array de cabeçalho.

- Retorno:**

**Nenhum**

  
*Importante* Na exportação das informações para o Microsoft Excel deve-se atentar para as colunas que possuírem informações alfanuméricas mas que contém apenas números. Para que estas colunas sejam exibidas corretamente deve ser acrescentado um caractere especial no início da string de forma que o Microsoft Excel as reconheça como texto e não como numéricas.

## **Exemplo de exportação para o Microsoft Excel utilizando a função DlgToExcel()**

```
#include "protheus.ch"

/*
+-----+
| Função | GExpExcel      | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Descrição | EXPORTACAO PARA EXCEL GENERICO |      |
+-----+
| Uso       | Curso ADVPL |      |
+-----+
 */

User Function GExpExcel()

Local aCabExcel    :={}
Local aItensExcel   :={}

// AADD(aCabExcel, {"TITULO DO CAMPO", "TIPO", NTAMANHO, NDECIMAIS})
AADD(aCabExcel, {"A1_FILIAL", "C", 02, 0})
AADD(aCabExcel, {"A1_COD", "C", 06, 0})
AADD(aCabExcel, {"A1_LOJA", "C", 02, 0})
AADD(aCabExcel, {"A1_NOME", "C", 40, 0})
AADD(aCabExcel, {"A1_MCOMPRA", "N", 18, 2})

MsgRun("Favor Aguardar.....", "Selecionando os Registros",;
       {|| GProcItens(aCabExcel, @aItensExcel)})

MsgRun("Favor Aguardar.....", "Exportando os Registros para o Excel",;
       {|| DlgToExcel({{"GETDADOS",;
                      "POSIÇÃO DE TÍTULOS DE VENDOR NO PERÍODO",;
                      aCabExcel,aItensExcel}})})

Return

/*
+-----+
| Função | GProcItens     | Autor | Arnaldo R. Junior | Data |      |
+-----+
| Uso       | Curso ADVPL |      |
+-----+
 */

Static Function GProcItens(aHeader, aCols)

Local aItem
Local nX

DbSelectArea("SA1")
DbSetOrder(1)
DbGotop()
```

**Continuação:**

```
While SA1->(!EOF())  
  
    aItem := Array(Len(aHeader))  
  
    For nX := 1 to Len(aHeader)  
        IF aHeader[nX][2] == "C"  
            aItem[nX] := CHR(160)+SA1->&(aHeader[nX][1])  
        ELSE  
            aItem[nX] := SA1->&(aHeader[nX][1])  
        ENDIF  
    Next nX  
  
    AADD(aCols,aItem)  
    aItem := {}  
    SA1->(dbSkip())  
  
End  
  
Return
```

**Anotações**

---

---

---

---

## APÊNDICES

### Relação de imagens para aplicações visuais

	AFASTAME		BPMSDOCI		COLTOT
	AFASTAMENTO		BPMSEDT1		CONTAINR
	ALT_CAD		BPMSEDT2		DBG05
	AMARELO		BPMSEDT3		DBG06
	ANALITICO		BPMSEDT4		DBG09
	ANALITIC		BPMREL A		DBG3
	AGENDA		BPMSTASK1		DESTINOS
	ALTERA		BPMSTASK2		DESTINOS2
	AREA		BPMSTASK3		DISABLE
	ASIENTOS		BPMSTASK4		DISCAGEM
	AUTOM		BR_AMARELO		DOWN
	BAIXATIT		BR_AZUL		E5
	BAR		BR_AZUL_OCEAN		EDITABLE
	BMPCALEN		BR_CINZA		EXCLUIR
	BMPEMERG		BR_LARANJA		FILTRO
	BMPGROUP		BR_MARROM		FINAL
	BMPINCLUIR		BR_PRETO		FOLDER10
	BMPPERG		BR_VERDE		FOLDER11
	BMPPOST		BR_VERDE_OCEAN		FOLDER12
	BMPTABLE		BR_VERMELHO		FOLDER14
	BMPTRG		BR_VERMELHO_OCEAN		FOLDER5
	BMPUSER		BUDGET		FOLDER6
	BMPVISUAL		BUDGETY		FOLDER7
	BONUS		CADEADO		GEOROTA

	BOTTOM		CALCULADORA		GRAF2D
	BPMSDOC		CANCEL		GRAF3D
	BPMSDOCA		CHAVE2		HISTORIC
	BPMSDOCE		CHECKED		INSTRUIME
	IMPRESSAO		PCO_ITALT		PMSSETATOP
	LBNO		PCO_ITEXC		PMSSETAUP
	LBOK		PCOCO		PMSTASK1
	LBTIK		PCOCUBE		PMSTASK2
	LEFT		PCOFCANCEL		PMSTASK3
	LINE		PCOFCOK		PMSTASK4
	LIQCHECK		PENDENTE		PMSUSER
	LJPRECO		PESQUISA		PMSZOOMIN
	LOCALIZA		PGNEXT		PMSZOOMOUT
	LUPA		PGPREV		POSCLI
	MAQFOTO		PMSCOLOR		PRECO
	MATERIAL		PMSEDT3		PREV
	METAS_BAIXO_16		PMSEDT4		PRINT03
	METAS_BAIXO_LEG		PMSEXCEL		PRODUTO
	METAS_CIMA_16		PMSEXPALL		RECALC
	METAS_CIMA_LEG		PMSEXP CMP		RECORTAR
	MSGHIGH		PMSMAIS		RIGHT
	MSVISIO		PMSMATE		RPMNEW
	NEXT		PMSMENOS		RPMSAVE
	NOTE		PMSPESQ		S4SB014N
	NOVACELULA		PMSPRINT		S4WB001N
	OBJETIVO		PMSRELA		S4WB005N

	OK		PMSRRFSH		S4WB006N
	ORDEM		PMSSETABOT		S4WB007N
	PARAMETROS		PMSSETADIR		S4WB008N
	PCO_COINC		PMSSETADOWN		S4WB009N
	PCO_CONOR		PMSSETAESQ		S4WB010N
	S4WB011N		WEB		CARAGANEW
	S4WB014B		WFCHK		CARGASEQ
	S4WB016N		WFUNCHK		CCTCALC
	SALVAR		ADDCONTAINER		CHAT
	SDUIMPORT		ADICIONAR_001		CHAT1
	SDUPACK		ARMAZEM		CHAT2
	SDUPROPR		ATALHO		CHAT3
	SDUSETDEL		AVGARMAZEM		CHECK
	SDUSOFTSEEK		AVGBOX1		CHECKOK
	SHORTCUTDEL		AVGLBPAR1		CLOCK01
	SHORTCUTEDIT		AVGOIC1		CLOCK02
	SHORTCUTMINUS		AVIAO		CLOCK03
	SHORTCUTNEW		AZUL		CLOCK04
	SHORTCUTPLUS		BALANCA		DEVOLNF
	SIMULACA		BGCOLOR		COBROWSR
	SIMULACAO		BMPPARAM		COLFORM
	SUGESTAO		BMPCONS		COMPTITL
	SUMARIO		BMPCPO		COMSOM
	SVM		BMPDEL		CRITICA
	TK_VERTIT		BR_BRANCO		COPYUSER
	UNCHECKED		BRANCO		CTBLANC

	UP		BR_CANCEL		CTBREPLA
	USER		BR_MARRON		DBG07
	VCDOWN		BR_PINK		DELWEB
	VCUP		BTCALC		COLOR
	VENDEDOR		BTPESQ		DBG12
	VERNOTA		CARGA		DBG10
	DEPENDENTES		F7_VERM		F14_PINK
	GEO		F8_NULL		F14_PRET
	EDITWEB		F10_AMAR		F14_VERD
	EMPILHADEIRA		F10_AZUL		F14_VERM
	ENABLE		F10_CINZ		FÉRIAS
	ESCALA		F10_LARA		FILTRO1
	ESTOMOVI		F10_MARR		FOLDER8
	F5_AZUL		F10_NULL		FOLDER13
	F5_NULL		F10_PINK		FOLDER15
	F5_VERD		F10_PRET		FORM
	F5_VERM		F10_VERD		FRCOLOR
	F6_NULL		F10_VERM		FRTOFFLINE
	F5_AMAR		F11_NULL		FRTONLINE
	F5_CINZ		F12_AMAR		GEO
	F5_LARA		F12_AZUL		GEOEMAIL
	F5_MARR		F12_CINZ		GEOTRECHO
	F5_PINK		F12_LARA		GERPROJ
	F5_PRET		F12_MARR		GLOBO
	F7_AMAR		F12_PINK		IC_17
	F7_AZUL		F12_PRET		INSTRUIME

	F7_CINZ		F12_VERD		LANDSCAPE
	F7_LARA		F12_VERM		LIGHTBLU
	F7_MARR		F14_AMAR		MDIHELP
	F7_NULL		F14_AZUL		MDILOGOFF
	F7_PINK		F14_CINZ		MDIRUN
	F7_PRET		F14_LARA		MDISPOOL
	F7_VERD		F14_MARR		MEDEXT
	MENURUN		RPMCABEC		SDUFIELDS
	MPWIZARD		RPMCPO		SDUFIND
	NCO		RPMDES		SDUGOTO
	NEWWEB		RPMFORM		SDUNEW
	NOCONNECT		RPMFUNC		SDUOPEN
	NOCHECKED		RPMGROUP		SDUOPENIDX
	NOMEDICA		RPMIMP		SDUORDER
	NORMAS		RPMIMPORT		SDURECALL
	OPEN		RPMNEW2		SDUREPL
	OPERACAO		RPMOPEN		SDUSEEK
	OUTLOOK		RPMPERG		SDUSTRUCT
	PAPEL_ESCRITO		RPMTABLE		SDUSUM
	PEDIDO		S4WB004N		SDUZAP
	PIN		S4WB013N		SEMSOM
	PMSINFO		S4WB014A		SOLICITA
	PREDIO		SALARIOS		SSFONTES
	PRINT02		SAVECLOCK		TAB1
	PROCESSA		SDUADDTBL		TABPRICE
	PRODUT2		SDUAPPEND		TEXTBOLD

	PROJETPMS		SDUCLOSE		TEXTCENTER
	PRTETQ		SDUCLOSEIDX		XCLOSE
	QMT_COND		SDUCOPYTO		TEXTITALIC
	QMT_NO		SDUCOUNT		TEXTJUSTIFY
	QMT_OK		SDUCREATEIDX		TEXTLEFT
	RESPADEX		SDUDELETE		TEXTRIGHT
	RESPONSA		SDUDRPTBL		TEXTUNDERLINE
	ROSA		SDUERASE		TK_ALTFIN
	TK_CLIFIN		BPMSEDT3A		GCT_NEW
	TK_FIND		BPMSEDT3E		INVOICE1
	TK_FONE		BPMSEDT3I		MSGGROUP
	TK_HISTORY		BPMSEDT4A		MSGHIGH
	TK_NOVO		BPMSEDT4E		PCO_COALT
	TK_REFRESH		BPMSEDT4I		PCO_COEXC
	TPOPAGTO1		BPMSREC		PCO_ITINC
	UPDWARNING		BPMSRECA		PCOCOLA
	UPDERROR		BPMSRECE		PCOCOPY
	UPDINFORMATION		BPMSRECI		PCOEDIT
	VERDE		BPMSRELAA		PCOFX
	VERMELHO		BPMSRELAE		PCOLOCK
	VERMESCOIRO		BPMSRELAI		PEDIDO2
	WATCH		BPMSTSK1A		PEDIDO2_MDI
	CLIENTE		BPMSTSK1E		PGRSAVE
	ACAO		BPMSTSK1I		PMSAPONT
	BOXBOM1		BPMSTSK2A		PMSCANC
	BOXBOM2		BPMSTSK2E		PMSCOLUMN

	BOXBOM3		BPMSTSK2I		PMSCONS
	BOXBOM4		BPMSTSK3A		PMSCUSTO
	BOXBOM5		BPMSTSK3E		PMSDATE
	BPMSEDT1A		BPMSTSK3I		PMSESTRU
	BPMSEDT1E		BPMSTSK4A		PMSEXCEL
	BPMSEDT1I		BPMSTSK4E		PMSEXEC
	BPMSEDT2A		BPMSTSK4I		PMSEXPEXC
	BPMSEDT2E		ENGRENAGEM2		PMSFILTER
	BPMSEDT2I		GCT_EDIT		PMSGRAFH
	PMSNEXT		MSGFORWD		UNSELECTALL
	PMSOPCAO		MSGREPLY		BSTART
	PMSPESQ		OMSDIVIDE		BTURNSHARPLEFT
	PMSPREV		PMSUPDOWN		BTURNSHARPRIGHT
	PMSPREVIO		SHAPE01		ENGRENAGEM
	PMSPRINT		SHAPE02		PAGEDOWN
	PMSPROG		SHAPE03		PAGEUP
	PMSSUPALOC		SHAPE04		SELECT
	PMSTOOLS		SHAPE05		SELECTALL
	PMSUSERP		TRIDOWN		BBEARRIGHT
	REFRESH		WORD		BEND
	SHAPE06		ROTEIRO		BKEEPLEFT
	TRILEFT		CLIPS_PQ		BKEEPRIGHT
	TRIRIGHT		NEXT_PQ		BLEFT
	VINCULA1		RELACIONAMENTO_DIREIRA_PQ		BRIGHT
	BAHEAD		PREV_PQ		CANCEL
	BBEARLEFT		UP.GIF		CONFIRM

	DOWN		GOTOP		INVERTSELECTION
	GOBOTTOM				

## LISTAS DE EXERCÍCIOS



Exercícios

01

Implementar a classe Aluno com os métodos New(), Inscrever() e Avaliar().



Anotações

---

---

---

---

02

Desenvolver um diálogo para interagir com a classe Aluno desenvolvida no exercício anterior.



Anotações

---

---

---

---

03

Implementar uma Enchoice para o cadastro de produtos (SB1).



Anotações

---

---

---

---

**04**

Converter o cadastro desenvolvido no exercício anterior para que o mesmo utilize o objeto MsMGet().



*Anotações*

---

---

---

---

**05**

Desenvolver uma interface de cadastro que combine os objetos MsMGet e MsNewGetDados causando o efeito de uma Modelo3().



*Anotações*

---

---

---

---

**06**

Adicionar na interface de cadastro desenvolvida no exercício anterior uma EnchoiceBar() com um botão definido pelo usuário que exiba o AxCadastro de um item utilizado na interface.



*Anotações*

---

---

---

---

**07**

Converter o fonte DIALOG\_OBJETOS.PRW para sintaxe orientada à objetos.



*Anotações*

---

---

---

---

**08**

Desenvolver uma MaWndBrowse utilizando um arquivo temporário.



*Anotações*

---

---

---

---

**09**

Implementar uma funcionalidade com múltiplos TwBrowses que permita pesquisar clientes e visualizar os títulos a receber dos mesmos.



*Anotações*

---

---

---

---

**10**

Implementar um objeto TFolder para montar um rodapé para uma interface que contenha MsMGet() e MsNewGetDados().



*Anotações*

---

---

---

---

**11**

Implementar a MsAdvSize() e MsObjSize() no fonte desenvolvido no exercício anterior.



*Anotações*

---

---

---

---

**12**

Desenvolver uma MaWndBrowse utilizando um Alias gerado pela função TcGenQry().



**Anotações**

---

---

---

---

**13**

Desenvolver uma MaWndBrowse utilizando um Alias gerado pela função TcGenQry() para um banco de dados externo a aplicação ERP.



**Anotações**

---

---

---

---

**14**

Desenvolver uma aplicação que execute a Stored Procedure padrão de reprocessamento de saldos contábeis.



**Anotações**

---

---

---

---

**15**

Implementar no fonte WndSqlConn desenvolvido anteriormente as funções TcInclui(), TcAltera(), TcExclui() com o uso das funções TEnchoice() e TcSqlExec().



**Anotações**

---

---

---

---

**16**

Implementar uma função que realize o envio de um texto simples para uma conta de e-mail @totvs, que permita sua fácil identificação.



*Anotações*

---

---

---

---

**17**

Implementar uma rotina que realize o recebimento do e-mail enviado anteriormente para a conta de e-mail @totvs.



*Anotações*

---

---

---

---

**18**

Implementar uma rotina automática para gravar as informações no cadastro de clientes (MATA030).



*Anotações*

---

---

---

---

## **Projeto: Avaliação prática do treinamento de ADVPL Avançado**

### **Objetivos do projeto**

- Contemplar o conteúdo do curso e estimular a prática da programação utilizando a linguagem ADVPL;
- Acrescentar um grau de desafio ao treinamento;

### **Regras para entrega e apresentação**

- Deverá ser entregue na data estabelecida pelo instrutor, contendo os fontes e demais arquivos necessários. A avaliação do projeto será realizada após a entrega, no prazo estabelecido pelo instrutor;
- Pode ser realizado em grupos de até 03 pessoas. Caso existam alunos sem grupos, estes poderão integrar equipes já formadas com o total de membros estabelecido, mediante aprovação do instrutor;

### **Avaliação do projeto**

- Mesmo peso da prova teórica, podendo substituí-la integralmente.
- Serão considerados os quesitos:
  - Funcionalidades
  - Clareza da programação
  - Apresentação
  - Otimização de técnicas e recursos apresentados no treinamento

### **Funcionalidades que compõe o projeto**

Desenvolver uma aplicação utilizando a linguagem ADVPL que contemple as seguintes funcionalidades:

<b>Projeto 01</b>	Desenvolver uma aplicação de cadastro utilizando a classe Aluno e os métodos disponíveis para a mesma.
<b>Projeto 02</b>	Converter os fontes do pacote de oficina de programação para orientação a objetos.
<b>Projeto 03</b>	Implementar uma aplicação ADVPL que consulte e atualize dados em um banco de interface externo ao Protheus.

# REFERÊNCIAS BIBLIOGRÁFICAS

## Referências bibliográficas

**Gestão empresarial com ERP**

Ernesto Haberkorn, 2006

**Programação Orientada a Objetos com C ++**

Ivan Luiz Marques Ricarte, 1996

**Modelagem e Projetos baseados em objetos**

James Rumbaugh, 1994 – 11ª. Edição

**Programação orientada à objetos no FiveWin**

Gilmer – FiveWin Brasil

**Oficina de Programação**

Robson Luiz Estefani Gonçalves

**Apostila de Treinamento - TReport**

Tânia Bronzeri

**Apostila de Treinamento - ADVPL**

Educação corporativa

**Apostila de Treinamento – Boas Práticas de Programação**

Inteligência Protheus e Fábrica de Software

**DEM – Documentação Eletrônica Microsiga**

Microsiga Software S.A.

**Materiais diversos de colaboradores Microsiga**

Colaboradores Microsiga

## Colaboradores

<b>Revisão 01</b>	<b>Data: 04.2008</b>
Arnaldo Raymundo Junior	CSA
Luis Akira Tamura	Educação Corporativa
Márcia Satiko Sasaki Tokura	ACR-N1_FRAMEW1
Patricia Lopes Legas	Educação Corporativa
Robson Luiz Estefani Gonçalves	Fábrica de Software
Sérgio Sueo Fuzinaka	Inteligência Protheus



# Protheus<sup>10</sup>

---

*Educação Corporativa*

---

## **Guia de Referência Rápida ADVPL Avançado**



Matriz – Av. Braz Leme, 1.717 – 02511-000 – São Paulo – SP – Brasil.  
Tel.: 55 (11) 3981-7001 [www.microsiga.com.br](http://www.microsiga.com.br)

## GUIA DE REFERÊNCIA RÁPIDA: Funções e Comandos ADVPL

Neste guia de referência rápida serão descritas as funções básicas da linguagem ADVPL, incluindo as funções herdadas da linguagem Clipper, necessárias ao desenvolvimento no ambiente ERP.

### Conversão entre tipos de dados

#### CTOD()

Realiza a conversão de uma informação do tipo caracter no formato "DD/MM/AAAA" para uma variável do tipo data.

- Sintaxe:** CTOD(cData)
- Parâmetros**

<b>cData</b>	Caracter no formato "DD/MM/AAAA"
--------------	----------------------------------

#### Exemplo:

```
cData := "31/12/2006"  
dData := CTOD(cData)  
  
IF dDataBase >= dData  
    MSGALERT("Data do sistema fora da competência")  
ELSE  
    MSGINFO("Data do sistema dentro da competência")  
ENDIF
```

#### CVALTOCHAR()

Realiza a conversão de uma informação do tipo numérico em uma string, sem a adição de espaços a informação.

- Sintaxe:** CVALTOCHAR(nValor)
- Parâmetros**

<b>nValor</b>	Valor numérico que será convertido para caractere.
---------------	--

#### Exemplo:

```
FOR nPercorridos := 1 to 10  
    MSGINFO("Passos percorridos: "+CvalToChar(nPercorridos))  
NEXT nPercorridos
```

## **DTOC()**

---

Realiza a conversão de uma informação do tipo data para em caracter, sendo o resultado no formato "DD/MM/AAAA".

- Sintaxe: DTOC(dData)**
- Parâmetros**

<b>dData</b>	Variável com conteúdo data
--------------	----------------------------

### **Exemplo:**

```
MSGINFO("Database do sistema: "+DTOC(dData))
```

---

## **DTOS()**

Realiza a conversão de uma informação do tipo data em um caracter, sendo o resultado no formato "AAAAMMDD".

- Sintaxe: DTOS(dData)**
- Parâmetros**

<b>dData</b>	Variável com conteúdo data
--------------	----------------------------

### **Exemplo:**

```
cQuery := "SELECT A1_COD, A1_LOJA, A1_NREDUZ FROM SA1010 WHERE "
cQuery += "A1_DULTCOM >=""+DTOS(dDataIni)+""
```

---

## **STOD()**

Realiza a conversão de uma informação do tipo caracter com conteúdo no formato "AAAAMMDD" em data.

- Sintaxe: STOD(sData)**
- Parâmetros**

<b>sData</b>	String no formato "AAAAMMDD"
--------------	------------------------------

### **Exemplo:**

```
sData := LERSTR(01,08) // Função que realiza a leitura de uma string de um txt previamente
          // aberto
dData := STOD(sData)
```

## **STR()**

---

Realiza a conversão de uma informação do tipo numérico em uma string, adicionando espaços à direita.

- Sintaxe: STR(nValor)**
- Parâmetros**

<b>nValor</b>	Valor numérico que será convertido para caractere.
---------------	--

### **Exemplo:**

```
FOR nPercorridos := 1 to 10
    MSGINFO("Passos percorridos: "+CvalToChar(nPercorridos))
NEXT nPercorridos
```

## **STRZERO()**

---

Realiza a conversão de uma informação do tipo numérico em uma string, adicionando zeros à esquerda do número convertido, de forma que a string gerada tenha o tamanho especificado no parâmetro.

- Sintaxe: STRZERO(nValor, nTamanho)**
- Parâmetros**

<b>nValor</b>	Valor numérico que será convertido para caractere.
<b>nTamanho</b>	Tamanho total desejado para a string retornada.

### **Exemplo:**

```
FOR nPercorridos := 1 to 10
    MSGINFO("Passos percorridos: "+CvalToChar(nPercorridos))
NEXT nPercorridos
```

## **VAL()**

Realiza a conversão de uma informação do tipo caracter em numérica.

- Sintaxe: VAL(cValor)**
- Parâmetros**

<b>cValor</b>	String que será convertida para numérico.
---------------	---

### **Exemplo:**

```
Static Function Modulo11(cData)
LOCAL L, D, P := 0
L := Len(cdata)
D := 0
P := 1
While L > 0
    P := P + 1
    D := D + (Val(SubStr(cData, L, 1)) * P)
    If P = 9
        P := 1
        End
        L := L - 1
    End
D := 11 - (mod(D,11))
If (D == 0 .Or. D == 1 .Or. D == 10 .Or. D == 11)
    D := 1
End
Return(D)
```



*Anotações*

---

---

---

---

---

## **Matemáticas**

### **ACOS()**

Função utilizada para calcular o valor do arco co-seno.

- Sintaxe:** ACOS(nValor)
- Parâmetros:**

<b>nValor</b>	Valor entre -1 e 1 de quem será calculado o Arco Co-Seno.
---------------	---

- Retorno:**

<b>Numérico</b>	Range de 0 a $\pi$ radianos.  Se o valor informado no parâmetro for menor que -1 ou maior que 1, <b>acos</b> retorna um valor indefinido por default $[+\infty, -\infty]$
-----------------	---

### **CEILING()**

Função utilizada para calcular o valor mais próximo possível de um valor nMax informado como parâmetro para a função.

- Sintaxe:** CEILING(nMax)
- Parâmetros**

<b>nMax</b>	Valor limite para análise da função, no formato floating-point.
-------------	---

- Retorno:**

<b>Numérico</b>	Valor do tipo double, representando o menor inteiro que é maior ou igual ao valor de nX. Não há retorno de erro na função.
-----------------	--

### **COS()**

Função utilizada para calcular o valor do co-seno ou co-seno hiperbólico.

**Importante:** Se  $x \geq 2^{63}$  ou  $x \leq -2^{63}$  ocorre perda significante na chamada da função COS().

- Sintaxe:** COS(nAngulo)
- Parâmetros:**

<b>nAngulo</b>	Valor que representa o ângulo em radianos.
----------------	--

- Retorno:**

<b>Numérico</b>	Valor que representa o co-seno ou co-seno hiperbólico do ângulo informado.
-----------------	--

**Situações inválidas:**

Entrada	Exceção apresentada	Significado da Exceção
$\pm \text{QNAN,IND}$	None	Sem Domínio
$\pm \infty (\cosf, \cos)$	INVALID	Sem Domínio
$x \geq 7.104760e+002 (\cosh, \coshf)$	INEXACT+OVERFLOW	OVERFLOW

## **LOG10()**

Função utilizada para calcular o logaritmo natural de um valor numérico, em base 10.

LOG10() é uma função numérica que calcula o logaritmo natural de um número. O logaritmo natural tem como base o valor 10. Devido ao arredondamento matemático, os valores retornados por LOG() podem não coincidir exatamente.

**Sintaxe: LOG10(nNatural)**

**Parâmetros:**

<b>nNatural</b>	Valor cujo o logaritmo deve ser encontrado.
-----------------	---

**Retorno:**

<b>Numérico</b>	A função retorna o logaritmo de nNatural se bem sucedidas. Se nNatural for negativo, estas funções retornam um indefinido, pelo defeito. Se nNatural for 0, retornam INF(infinito).
-----------------	---

## **SIN()**

Função utilizada para calcular o valor do seno ou seno hiperbólico. Devemos informar como parâmetro para a função um valor que representa o angulo em radianos.

**Importante:** Se  $x \geq 2^{63}$  ou  $x \leq -2^{63}$  ocorre perda significante na chamada da função SIN().

**Sintaxe: SIN(nAngulo)**

**Parâmetros:**

<b>nAngulo</b>	Valor do ângulo em radianos.
----------------	------------------------------

**Retorno:**

<b>Numérico</b>	Retorna o valor do seno do ângulo especificado.
-----------------	---

**Situações inválidas:**

Entrada	Exceção apresentada	Significado da Exceção
$\pm \text{QNAN,IND}$	None	Sem Domínio
$\pm \infty (\senf, \sen)$	INVALID	Sem Domínio
$x \geq 7.104760e+002 (\senh, \senhf)$	INEXACT+OVERFLOW	OVERFLOW

## **SQRT()**

Função utilizada para calcular a raiz quadrada de um número positivo.

- Sintaxe:** SQRT(nValor)  
 **Parâmetros:**

<b>nValor</b>	Um número positivo do qual será calculada a raiz quadrada.
---------------	--

- Retorno:**

<b>Numérico</b>	Retorna um valor numérico calculado com precisão dupla. A quantidade de casas decimais exibidas é determinada apenas por SET DECIMALS, sem importar a configuração de SET FIXED. Um número negativo <nValor> retorna zero.
-----------------	--

## **TAN()**

Função utilizada para calcular o valor da tangente ou tangente hiperbólica.

**Importante:** Se  $x \geq 2^{63}$  ou  $x \leq -2^{63}$  ocorre perda significante na chamada da função cos.

- Sintaxe:** TAN(nAngulo)  
 **Parâmetros:**

<b>nAngulo</b>	Valor do ângulo em radianos.
----------------	------------------------------

- Retorno:**

<b>Numérico</b>	Retorna o valor da tangente do ângulo especificado.
-----------------	---

- Situações inválidas:**

<b>Entrada</b>	<b>Exceção apresentada</b>	<b>Significado da Exceção</b>
$\pm \text{QNAN}, \text{IND}$	None	Sem Domínio
$\pm \infty$	INVALID	Sem Domínio



Anotações

---



---



---



---

## Análise de variáveis

### TYPE()

Determina o tipo do conteúdo de uma variável, a qual não foi definida na função em execução.

- Sintaxe: TYPE("cVariavel")**
- Parâmetros**

<b>"cVariavel"</b>	Nome da variável que se deseja avaliar, entre aspas ("").
--------------------	---

#### Exemplo:

```
IF TYPE("dDataBase") == "D"
    MSGINFO("Database do sistema: "+DTOC("dDataBase"))
ELSE
    MSGINFO("Variável indefinida no momento")
ENDIF
```

### VALTYPE()

Determina o tipo do conteúdo de uma variável, a qual não foi definida na função em execução.

- Sintaxe: VALTYPE(cVariavel)**
- Parâmetros**

<b>cVariavel</b>	Nome da variável que se deseja avaliar.
------------------	---

#### Exemplo:

```
STATIC FUNCTION GETTEXTO(nTamanho, cTitulo, cSay)
LOCAL cTexto      := ""
LOCAL nColF, nLargGet   := 0
PRIVATE oDlg
Default cTitulo     := "Tela para informar texto"
Default cSay        := "Informe o texto:"
Default nTamanho    := 1

nTamanho     := IIF(ValType(nTamanho) != "N",1,nTamanho) // Se o parâmetro foi passado
cTexto       := Space(nTamanho);nLargGet := Round(nTamanho * 2.5,0);
nColF        := Round(195 + (nLargGet * 1.75) ,0)

DEFINE MSDIALOG oDlg TITLE cTitulo FROM 000,000 TO 120,nColF PIXEL
@ 005,005 TO 060, Round(nColF/2,0) OF oDlg PIXEL
@ 010,010 SAY cSay SIZE 55, 7 OF oDlg PIXEL
@ 010,065 MSGET cTexto SIZE nLargGet, 11 OF oDlg PIXEL ;
Picture "@!" VALID !Empty(cTexto)
DEFINE SBUTTON FROM 030, 010 TYPE 1 ;
ACTION (nOpca := 1,oDlg:End()) ENABLE OF oDlg
DEFINE SBUTTON FROM 030, 040 TYPE 2 ;
ACTION (nOpca := 0,oDlg:End()) ENABLE OF oDlg
ACTIVATE MSDIALOG oDlg CENTERED
cTexto := IIF(nOpca==1,cTexto,"")
RETURN cTexto
```

## Manipulação de arrays

### AADD()

A função AADD() permite a inserção de um item em um array já existente, sendo que este item podem ser um elemento simples, um objeto ou outro array.

- Sintaxe: AADD(aArray, xItem)**
- Parâmetros**

<b>aArray</b>	Array pré-existente no qual será adicionado o item definido em xItem
<b>xItem</b>	Item que será adicionado ao array.

#### Exemplo:

```
aDados := {} // Define que a variável aDados é um array, sem especificar suas dimensões.  
aItem := {} // Define que a variável aItem é um array, sem especificar suas dimensões.  
  
AADD(aItem, cVariavel1) // Adiciona um elemento no array aItem de acordo com o cVariavel1  
AADD(aItem, cVariavel2) // Adiciona um elemento no array aItem de acordo com o cVariavel2  
AADD(aItem, cVariavel3) // Adiciona um elemento no array aItem de acordo com o cVariavel3  
  
// Neste ponto o array a Item possui 03 elementos os quais podem ser acessados com:  
// aItem[1] -> corresponde ao conteúdo de cVariavel1  
// aItem[2] -> corresponde ao conteúdo de cVariavel2  
// aItem[3] -> corresponde ao conteúdo de cVariavel3  
  
AADD(aDados,aItem) // Adiciona no array aDados o conteúdo do array aItem  
  
// Neste ponto, o array a aDados possui apenas um elemento, que também é um array  
// contendo 03 elementos:  
// aDados [1][1] -> corresponde ao conteúdo de cVariavel1  
// aDados [1][2] -> corresponde ao conteúdo de cVariavel2  
// aDados [1][3] -> corresponde ao conteúdo de cVariavel3  
  
AADD(aDados, aItem)  
AADD(aDados, aItem)  
  
// Neste ponto, o array aDados possui 03 elementos, aonde cada qual é um array com outros  
// 03 elementos, sendo:  
  
// aDados [1][1] -> corresponde ao conteúdo de cVariavel1  
// aDados [1][2] -> corresponde ao conteúdo de cVariavel2  
// aDados [1][3] -> corresponde ao conteúdo de cVariavel3  
  
// aDados [2][1] -> corresponde ao conteúdo de cVariavel1  
// aDados [2][2] -> corresponde ao conteúdo de cVariavel2  
// aDados [2][3] -> corresponde ao conteúdo de cVariavel3  
  
// aDados [3][1] -> corresponde ao conteúdo de cVariavel1  
// aDados [3][2] -> corresponde ao conteúdo de cVariavel2  
// aDados [3][3] -> corresponde ao conteúdo de cVariavel3
```

### **Exemplo (continuação):**

```
// Desta forma, o array aDados montando com uma estrutura de 03 linhas e 03 colunas, com  
// o conteúdo definido por variáveis externas, mas com a mesma forma obtida com o uso do  
// comando: aDados := ARRAY(3,3).
```

### **ACLONE()**

A função ACLONE() realiza a cópia dos elementos de um array para outro array integralmente.

- Sintaxe: AADD(aArray)**
- Parâmetros**

<b>aArray</b>	Array pré-existente que terá seu conteúdo copiado para o array especificado.
---------------	--

### **Exemplo:**

```
// Utilizando o array aDados utilizado no exemplo da função AADD()  
aItens := ACLONE(aDados)
```

```
// Neste ponto, o array aItens possui exatamente a mesma estrutura e informações do array  
// aDados.
```



*Importante*

Por ser uma estrutura de memória, um array não pode ser simplesmente copiado para outro array através de uma atribuição simples (":=").

Para mais informações sobre a necessidade de utilizar o comando ACLONE() verifique o tópico 6.1.3 – Cópia de Arrays.

### **ACOPY()**

Função de array que copia elementos do array aOrigem para array aDestino. O array destino aDestino já deve ter sido declarado e grande o bastante para conter os elementos que serão copiados. Se o array aOrigem contiver mais elementos, alguns dos elementos não serão copiados. ACOPY() copia os valores de todos os dados, incluindo valores nulos (NIL) e códigos de bloco.

Se um elemento for um subarray, o elemento correspondente no array aDestino, conterá o mesmo subarray. Portanto, ACOPY() não produzirá uma cópia completa de array multidimensionais.

- Sintaxe: ACOPY( aOrigem, aDestino , [ nInicio ], [ nQtde ], [ nPosDestino ] )**
- Parâmetros:**

<b>aOrigem</b>	é o array que contém os elementos a serem copiados.
<b>aDestino</b>	é o array que receberá a cópia dos elementos.
<b>nInicio</b>	indica qual o índice do primeiro elemento de <i>aOrigem</i> que será copiado. Se não for especificado, o valor assumido será 01.

<b>nQtde</b>	indica a quantidade de elementos a serem copiados a partir do array <i>aOrigem</i> . iniciando-se a contagem a partir da posição <i>nInicio</i> . Se <i>nQtde</i> não for especificado, todos os elementos do array <i>aOrigem</i> serão copiados, iniciando-se a partir da posição <i>nInicio</i> .
<b>nPosDestino</b>	é a posição do elemento inicial no array <i>aDestino</i> que receberá os elementos de <i>aOrigem</i> . Se não especificado, será assumido 01.

**Retorno:**

<b>aDestino</b>	referência ao array <i>aDestino</i> .
-----------------	---------------------------------------

**Exemplo:**

```
LOCAL nCount := 2, nStart := 1, aOne, aTwo
aOne := { 1, 1, 1 }
aTwo := { 2, 2, 2 }
ACOPY(aOne, aTwo, nStart, nCount)
// Result: aTwo is now { 1, 1, 2 }
```

## ADEL()

A função ADEL() permite a exclusão de um elemento do array. Ao efetuar a exclusão de um elemento, todos os demais são reorganizados de forma que a ultima posição do array passará a ser nula.

- Sintaxe: ADEL(aArray, nPosição)**  
 **Parâmetros**

<b>aArray</b>	Array do qual deseja-se remover uma determinada posição
<b>nPosição</b>	Posição do array que será removida

**Exemplo:**

```
// Utilizando o array aItens do exemplo da função ACLONE() temos:
ADEL(aItens,1) // Será removido o primeiro elemento do array aItens.

// Neste ponto, o array aItens continua com 03 elementos, aonde:
// aItens[1] -> antigo aItens[2], o qual foi reordenado como efeito da exclusão do item 1.
// aItens[2] -> antigo aItens[3], o qual foi reordenado como efeito da exclusão do item 1.
// aItens[3] -> conteúdo nulo, por se tratar do item excluído.
```

## **ADIR()**

Função que preenche os arrays passados com os dados dos arquivos encontrados, através da máscara informada. Tanto arquivos locais (Remote) como do servidor podem ser informados.

**Importante:** ADir é uma função obsoleta, utilize sempre Directory().

- Sintaxe:** **ADIR([ cArqEspec ], [ aNomeArq ], [ aTamanho ], [ aData ], [ aHora ], [ aAtributo ])**

- Parâmetros:**

<b>cArqEspec</b>	Caminho dos arquivos a serem incluídos na busca de informações. Segue o padrão para especificação de arquivos, aceitando arquivos no servidor Protheus e no Cliente. Caracteres como * e ? são aceitos normalmente. Caso seja omitido, serão aceitos todos os arquivos do diretório default ( *.* ).
<b>aNomeArq</b>	Array de Caractere. É o array com os nomes dos arquivos encontrados na busca. O conteúdo anterior do array é apagado.
<b>aTamanho</b>	Array Numérico. São os tamanhos dos arquivos encontrados na busca.
<b>aData</b>	Array de Datas. São as datas de modificação dos arquivos encontrados na busca.
<b>aHora</b>	Array de Caractere. São os horários de modificação dos arquivos encontrados. Cada elemento contém horário no formato: hh:mm:ss.
<b>aAtributos</b>	Array de Caractere. São os atributos dos arquivos, caso esse array seja passado como parâmetros, serão incluídos os arquivos com atributos de sistema e ocultos.

- Retorno:**

<b>nArquivos</b>	Quantidade de arquivos encontrados.
------------------	-------------------------------------

### **Exemplo:**

```
LOCAL aFiles[ADIR("*.TXT")]
ADIR("*.TXT", aFiles)
AEVAL(aFiles, { |element| QOUT(element) })
```



*Anotações*

---

---

---

---

## **AFILL()**

Função de manipulação de arrays, que preenche os elementos do array com qualquer tipo de dado. Incluindo code-block. Esta função não deve ser usada para preencher um array com outro array.

- Sintaxe:** AFILL( *aDestino* , *xExpValor*, [ *nInicio* ], [ *nQuantidade* ] )
- Parâmetros**

<b>aDestino</b>	É o onde os dados serão preenchidos.
<b>xExpValor</b>	É o dado que será preenchido em todas as posições informadas, não é permitida a utilização de arrays.
<b>nInicio</b>	É a posição inicial de onde os dados serão preenchidos, o valor padrão é 1.
<b>nCount</b>	Quantidade de elementos a partir de [ <i>nInicio</i> ] que serão preenchidos com <expValor>, caso não seja informado o valor será a quantidade de elementos até o final do array.

- Retorno:**

<b>aDestino</b>	Retorna uma referência para aDestino.
-----------------	---------------------------------------

### **Exemplo:**

```
LOCAL aLogic[3]
// Resultado: aLogic é { NIL, NIL, NIL }
AFILL(aLogic, .F.)
// Resultado: aLogic é { .F., .F., .F. }
AFILL(aLogic, .T., 2, 2)
// Resultado: aLogic é { .F., .T., .T. }
```

## **AINS()**

A função AINS() permite a inserção de um elemento no array especificado em qualquer ponto da estrutura do mesmo, diferindo desta forma da função AADD() a qual sempre insere um novo elemento ao final da estrutura já existente.

- Sintaxe:** AINS(*aArray*, *nPosicao*)
- Parâmetros**

<b>aArray</b>	Array pré-existente no qual desejasse inserir um novo elemento.
<b>nPosicao</b>	Posição na qual o novo elemento será inserido.

### **Exemplo:**

```
aAlunos := {"Edson", "Robson", "Renato", "Tatiana"}

AINS(aAlunos,3)
// Neste ponto o array aAlunos terá o seguinte conteúdo:
// {"Edson", "Robson", nulo, "Renato", "Tatiana"}
```



Similar ao efeito da função ADEL(), o elemento inserido no array pela função AINS() terá um conteúdo nulo, sendo necessário trata-lo após a realização deste comando.

## **ARRAY()**

A função Array() é utilizada na definição de variáveis de tipo array, como uma opção a sintaxe utilizando chaves ("{}").

- Sintaxe: Array(nLinhas, nColunas)**
- Parâmetros**

<b>nLinhas</b>	Determina o número de linhas com as quais o array será criado
<b>nColunas</b>	Determina o número de colunas com as quais o array será criado

### **Exemplo:**

```
aDados := Array(3,3) // Cria um array de três linhas, cada qual com 3 colunas.
```



O array definido pelo comando Array() apesar de já possuir a estrutura solicitada, não possui conteúdo em nenhum de seus elementos, ou seja:

`aDados[1]` -> array de três posições  
`aDados[1][1]` -> posição válida, mas de conteúdo nulo.

## **ASCAN()**

A função ASCAN() permite que seja identificada a posição do array que contém uma determinada informação, através da análise de uma expressão descrita em um bloco de código.

- Sintaxe: ASCAN(aArray, bSeek)**
- Parâmetros**

<b>aArray</b>	Array pré-existente no qual desejasse identificar a posição que contém a informação pesquisada.
<b>bSeek</b>	Bloco de código que configura os parâmetros da busca a ser realizada.

### **Exemplo:**

```
aAlunos := {"Márcio", "Denis", "Arnaldo", "Patrícia"}
```

```
bSeek := { |x| x == "Denis"}
```

```
nPosAluno := aScan(aAlunos,bSeek) // retorno esperado → 2
```



Dica

Durante a execução da função aScan, a variável "x" receberá o conteúdo o item que está posicionado no momento, no caso aAlunos[x]. Como aAlunos[x] é uma posição do array que contém o nome do aluno, "x" poderia ser renomeada para cNome, e a definição do bloco bSeek poderia ser re-escrita como:

```
bSeek := {|cNome| cNome == "Denis"}
```



Dica

Na definição dos programas é sempre recomendável utilizar variáveis com nomes significativos, desta forma os blocos de código não são exceção.

Sempre opte por analisar como o bloco de código será utilizado e ao invés de "x", "y" e similares, defina os parâmetros com nomes que representem seu conteúdo. Será mais simples o seu entendimento e o entendimento de outros que forem analisar o código escrito.

## **ASCANX()**

Função utilizada para varrer um vetor procurando um valor especificado, operando de forma similar a função ASCAN.

A diferença fundamental da função ASCANX é que esta função recebe um segundo parâmetro em seu code-block representando o índice do array.

- Sintaxe:** ASCANX ( < xDestino > , < bSeek > , [ nInicio ] , [ nCont ] )
- Parâmetros:**

<b>xDestino</b>	Representa o objeto a ser varrido pela função, pode ser atribuído ao parâmetro um array um Objeto.
<b>bSeek</b>	Representa o valor que será pesquisado, podendo ser um bloco de código.
<b>nInicio</b>	Representa o elemento a partir do qual terá inicio a pesquisa, quando este argumento não for informado o valor default será 1.
<b>nCont</b>	Representa a quantidade de elementos que serão pesquisados a partir da posição inicial, quando este argumento não for informado todos elementos do array serão pesquisados.

### **Exemplo.:**

```
nPos := aScanX( ARRAY, { |X,Y| X[1] == cNome .OR. y<=100} )
```



Dica

No código demonstrado acima, note a inclusão no code-block do Y, onde a função irá terminar sua execução em 3 condições:

- 1) Até encontrar o elemento no ARRAY com a ocorrência cNome, retornando a posição desse elemento.
- 2) Essa é novidade, ASCANX irá verificar o Array até a posição 100.
- 3) O elemento cNome não foi encontrado no ARRAY e a condição de Y até 100 não satisfaz, pois o array é menor do que 100 posições!



Dica

Como ASCAN() que utiliza o operador (=) para comparações, a função ASCANX() também é case sensitive, no caso os elementos procurados devem ser exatamente igual.

## ASIZE()

A função ASIZE permite a redefinição da estrutura de um array pré-existente, adicionando ou removendo itens do mesmo.

- Sintaxe: ASIZE(aArray, nTamanho)**
- Parâmetros**

<b>aArray</b>	Array pré-existente que terá sua estrutura redimensionada.
<b>nTamanho</b>	Tamanho com o qual deseja-se redefinir o array. Se o tamanho for menor do que o atual, serão removidos os elementos do final do array, já se o tamanho for maior do que o atual serão inseridos itens nulos ao final do array.

### Exemplo:

```
// Utilizando o array aItens, o qual teve um elemento excluído pelo uso da função ADEL()  
ASIZE(aItens,Len(aItens-1))  
  
// Neste ponto o array aItens possui 02 elementos, ambos com conteúdos válidos.
```



Dica

Utilizar a função ASIZE() após o uso da função ADEL() é uma prática recomendada e evita que seja acessada uma posição do array com um conteúdo inválido para a aplicação em uso.

## **ASORT()**

A função ASORT() permite que os itens de um array sejam ordenados a partir de um critério pré-estabelecido.

- Sintaxe: ASORT(aArray, nInicio, nItens, bOrdem)**
- Parâmetros**

<b>aArray</b>	Array pré-existente que terá seu conteúdo ordenado através de um critério estabelecido.
<b>nInicio</b>	Posição inicial do array para início da ordenação. Caso não seja informado, o array será ordenado a partir de seu primeiro elemento.
<b>nItens</b>	Quantos itens, a partir da posição inicial deverão ser ordenados. Caso não seja informado, serão ordenados todos os elementos do array.
<b>bOrdem</b>	Bloco de código que permite a definição do critério de ordenação do array. Caso bOrdem não seja informado, será utilizado o critério ascendente.

Um bloco de código é basicamente uma função escrita em linha. Desta forma sua estrutura deve “suportar” todos os requisitos de uma função, os quais são através da análise e interpretação de parâmetros recebidos, executar um processamento e fornecer um retorno.

Com base nesse requisito, pode-se definir um bloco de código com a estrutura abaixo:



Dica

bBloco := { |xPar1, xPar2, ... xParZ| Ação1, Ação2, AçãoZ } , aonde:

|| -> define o intervalo onde estão compreendidos os parâmetros  
Ação Z-> expressão que será executadas pelo bloco de código  
Ação1... AçãoZ -> intervalo de expressões que serão executadas pelo bloco de código, no formato de lista de expressões.

Retorno -> resultado da ultima ação executada pelo bloco de código, no caso AçãoZ.

Para maiores detalhes sobre a estrutura e utilização de blocos de código consulte o tópico 6.2 – Listas de Expressões e Blocos de código.

### **Exemplo 01 – Ordenação ascendente**

```
aAlunos := { "Mauren", "Soraia", "Andréia"}
```

```
aSort(aAlunos)
```

```
// Neste ponto, os elementos do array aAlunos serão {"Andréia", "Mauren", "Soraia"}
```

### **Exemplo 02 – Ordenação descendente**

```
aAlunos := { "Mauren", "Soraia", "Andréia"}  
bOrdem := {|x,y| x > y }  
  
// Durante a execução da função aSort(), a variável "x" receberá o conteúdo do item que está  
// posicionado. Como o item que está posicionado é a posição aAlunos[x] e aAlunos[x] ->  
// string contendo o nome de um aluno, pode-se substituir "x" por cNomeAtu.  
// A variável "y" receberá o conteúdo do próximo item a ser avaliado, e usando a mesma  
// analogia de "x", pode-se substituir "y" por cNomeProx. Desta forma o bloco de código  
// bOrdem pode ser re-escrito como:  
  
bOrdem := {|cNomeAtu, cNomeProx| cNomeAtu > cNomeProx}  
  
aSort(aAlunos,,bOrdem)  
  
// Neste ponto, os elementos do array aAlunos serão {"Soraia", "Mauren", "Andréia"}
```

### **ATAIL()**

ATAIL() é uma função de manipulação de array que retorna o último elemento de um array. Ela deve ser usada em substituição da seguinte construção: aArray [LEN( aArray )]

- Sintaxe:** ATAIL( aArray )
- Parâmetros:**

<b>aArray</b>	É o array de onde será retornado o último elemento.
---------------	---

- Retorno:**

<b>nÚltimo</b>	Número do último elemento do array.
----------------	-------------------------------------

### **Exemplo:**

```
aArray := {"a", "b", "c", "d"}  
ATAIL(aArray) // Resultado: d
```



*Anotações*

---

---

---

---

## Manipulação de blocos de código

### EVAL()

A função EVAL() é utilizada para avaliação direta de um bloco de código, utilizando as informações disponíveis no mesmo de sua execução. Esta função permite a definição e passagem de diversos parâmetros que serão considerados na interpretação do bloco de código.

- Sintaxe:** **EVAL(bBloco, xParam1, xParam2, xParamZ)**
- Parâmetros**

<b>bBloco</b>	Bloco de código que será interpretado.
<b>xParamZ</b>	Parâmetros que serão passados ao bloco de código. A partir da passagem do bloco, todos os demais parâmetros da função serão convertidos em parâmetros para a interpretação do código.

#### Exemplo:

```
nInt := 10
bBloco := {|N| x:= 10, y:= x*N, z:= y/(x*N)}
nValor := EVAL(bBloco, nInt)
// O retorno será dado pela avaliação da ultima ação da lista de expressões, no caso "z".
// Cada uma das variáveis definidas em uma das ações da lista de expressões fica disponível
// para a próxima ação.
// Desta forma temos:
// N → recebe nInt como parâmetro (10)
// X → tem atribuído o valor 10 (10)
// Y → resultado da multiplicação de X por N (100)
// Z → resultado a divisão de Y pela multiplicação de X por N ( 100 / 100 ) → 1
```

### DBEVAL()

A função DBEval() permite que todos os registro de uma determinada tabela sejam analisados e para cada registro será executado o bloco de código definido.

- Sintaxe:** **Array(bBloco, bFor, bWhile)**
- Parâmetros**

<b>bBloco</b>	Bloco de código principal, contendo as expressões que serão avaliadas para cada registro do alias ativo.
<b>bFor</b>	Condição para continuação da análise dos registros, com o efeito de uma estrutura For ... Next.
<b>bWhile</b>	Condição para continuação da análise dos registros, com o efeito de uma estrutura While ... End

**Exemplo 01:**

// Considerando o trecho de código abaixo:

```
dbSelectArea("SX5")
dbSetOrder(1)
dbGotop()

While !Eof() .And. X5_FILIAL == xFilial("SX5") .And.; X5_TABELA <= mv_par02
    nCnt++
    dbSkip()
End
```

**// O mesmo pode ser re-escrito com o uso da função DBEVAL():**

```
dbEval( { |x| nCnt++ },,{||X5_FILIAL==xFilial("SX5") .And. X5_TABELA<=mv_par02})
```

**Exemplo 02:**

// Considerando o trecho de código abaixo:

```
dbSelectArea("SX5")
dbSetOrder(1)
dbGotop()

While !Eof() .And. X5_TABELA == cTabela
    AADD(aTabela,{X5_CHAVE,Capital(X5_DESCRI)})
    dbSkip()
End
```

**// O mesmo pode ser re-escrito com o uso da função DBEVAL():**

```
dbEval({|| AADD(aTabela,{X5_CHAVE,Capital(X5_DESCRI)})},,{|| X5_TABELA==cTabela})
```



*Importante*

Na utilização da função DBEVAL() deve ser informado apenas um dos dois parâmetros: bFor ou bWhile.



*Anotações*

---

---

---

---

## **AEVAL()**

A função AEVAL() permite que todos os elementos de um determinada array sejam analisados e para cada elemento será executado o bloco de código definido.

- Sintaxe: AEVAL(aArray, bBloco, nInicio, nFim)**
- Parâmetros**

<b>aArray</b>	Array que será avaliado na execução da função.
<b>bBloco</b>	Bloco de código principal, contendo as expressões que serão avaliadas para cada elemento do array informado.
<b>nInicio</b>	Elemento inicial do array, a partir do qual serão avaliados os blocos de código.
<b>nFim</b>	Elemento final do array, até o qual serão avaliados os blocos de código.

### **Exemplo 01:**

**Considerando o trecho de código abaixo:**

```
AADD(aCampos,"A1_FILIAL")
AADD(aCampos,"A1_COD")
SX3->(dbSetOrder(2))
For nX:=1 To Len(aCampos)
    SX3->(dbSeek(aCampos[nX]))
    AADD(aTitulos,AllTrim(SX3->X3_TITULO))
Next nX
```

**O mesmo pode ser re-escrito com o uso da função AEVAL():**

```
aEval(aCampos,{|x| SX3->(dbSeek(x)),IIF(Found(), AADD(aTitulos,;
AllTrim(SX3->X3_TITULO)))})
```



*Anotações*

---

---

---

---

## Manipulação de strings

### ALLTRIM()

Retorna uma string sem os espaços à direita e à esquerda, referente ao conteúdo informado como parâmetro.

A função ALLTRIM() implementa as ações das funções RTRIM ("right trim") e LTRIM ("left trim").

- Sintaxe: ALLTRIM(cString)**
- Parâmetros**

<b>cString</b>	String que será avaliada para remoção dos espaços a direita e a esquerda.
----------------	---

#### **Exemplo:**

```
cNome := ALLTRIM(SA1->A1_NOME)  
  
MSGINFO("Dados do campo A1_NOME:" + CRLF  
"Tamanho:" + CVALTOCHAR(LEN(SA1->A1_NOME)) + CRLF  
"Texto:" + CVALTOCHAR(LEN(cNome)))
```

### ASC()

Converte uma informação caractere em seu valor de acordo com a tabela ASCII.

- Sintaxe: ASC(cCaractere)**
- Parâmetros**

<b>cCaractere</b>	Caracter que será consultado na tabela ASCII.
-------------------	---

#### **Exemplo:**

```
USER FUNCTION NoAcento(Arg1)  
Local nConta := 0  
Local cLetra := ""  
Local cRet := ""  
Arg1 := Upper(Arg1)  
For nConta:= 1 To Len(Arg1)  
    cLetra := SubStr(Arg1, nConta, 1)  
    Do Case  
    Case (Asc(cLetra) > 191 .and. Asc(cLetra) < 198) .or.;  
        (Asc(cLetra) > 223 .and. Asc(cLetra) < 230)  
        cLetra := "A"  
    Case (Asc(cLetra) > 199 .and. Asc(cLetra) < 204) .or.;  
        (Asc(cLetra) > 231 .and. Asc(cLetra) < 236)  
        cLetra := "E"
```

**Exemplo (continuação):**

```
Case (Asc(cLetra) > 204 .and. Asc(cLetra) < 207) .or.;  
(Asc(cLetra) > 235 .and. Asc(cLetra) < 240)  
    cLetra := "I"  
  
Case (Asc(cLetra) > 209 .and. Asc(cLetra) < 215) .or.;  
(Asc(cLetra) == 240) .or. (Asc(cLetra) > 241 .and. Asc(cLetra) < 247)  
    cLetra := "O"  
  
Case (Asc(cLetra) > 216 .and. Asc(cLetra) < 221) .or.;  
(Asc(cLetra) > 248 .and. Asc(cLetra) < 253)  
    cLetra := "U"  
  
Case Asc(cLetra) == 199 .or. Asc(cLetra) == 231  
    cLetra := "C"  
  
EndCase  
  
    cRet := cRet+cLetra  
  
Next  
  
Return UPPER(cRet)
```

**AT()**

Retorna a primeira posição de um caracter ou string dentro de outra string especificada.

- Sintaxe: AT(cCaractere, cString )**
- Parâmetros**

<b>cCaractere</b>	Caractere ou string que se deseja verificar
<b>cString</b>	String na qual será verificada a existência do conteúdo de cCaractere.

**Exemplo:**

```
STATIC FUNCTION NOMASCARA(cString,cMascara,nTamanho)
```

```
LOCAL cNoMascara := ""  
LOCAL nX := 0  
  
IF !Empty(cMascara) .AND. AT(cMascara,cString) > 0  
    FOR nX := 1 TO Len(cString)  
        IF !(SUBSTR(cString,nX,1) $ cMascara)  
            cNoMascara += SUBSTR(cString,nX,1)  
        ENDIF  
    NEXT nX  
    cNoMascara := PADR(ALLTRIM(cNoMascara),nTamanho)  
ELSE  
    cNoMascara := PADR(ALLTRIM(cString),nTamanho)  
ENDIF  
  
RETURN cNoMascara
```

## **BITON()**

---

Função utilizada para ligar determinados bits de uma String passada por parâmetro para a função. Além da string à ser alterada, a função também recebe como parâmetro um numérico que indica o bit de inicio a ser alterado, um numérico que indica a quantidade de bits a serem alterados(ligados) e o tamanho da string passada.

- Sintaxe:** **BITON ( < cValue > , < nBitIni > , < nBitEnd > , < nStrLen > )**
- Parâmetros**

<b>cValue</b>	String no qual desejamos ligar os bits.
<b>nBitIni</b>	Indica a partir de qual bit, começará a ser ligados os bits na String
<b>nBitEnd</b>	Indica a quantidade de bits que serão ligados a partir do inicio.
<b>nStrLen</b>	Representa o tamanho da String passada para a função.

## **CAPITAL()**

---

Função que avalia a string passada como parâmetro alterando a primeira letra de cada palavra para maiúscula e as demais letras como minúsculas.

- Sintaxe:** **CAPITAL(cFrase)**

- Parâmetros:**

<b>cFrase</b>	String a ser avaliada
---------------	-----------------------

- Retorno:**

<b>String</b>	Conteúdo da string original com as modificações necessárias para atender a condição da função.
---------------	--

## **CHR()**

---

Converte um valor número referente a uma informação da tabela ASCII no caractere que esta informação representa.

- Sintaxe:** **CHR(nASCII)**
- Parâmetros**

<b>nASCII</b>	Código ASCII do caractere
---------------	---------------------------

### **Exemplo:**

```
#DEFINE CRLF CHR(13)+CHR(10) // FINAL DE LINHA
```

## **DESCEND()**

---

Função de conversão que retorna a forma complementada da expressão string especificada. Esta função normalmente é utilizada para a criação de indexadores em ordem decrescente

- Sintaxe:** DESCEND ( < cString > )
- Parâmetros:**

<b>cString</b>	Corresponde à seqüência de caracteres a ser analisada.
----------------	--

- Retorno:**

<b>Caracter</b>	String complementada da string analisada.
-----------------	---

### **Exemplo:**

```
// Este exemplo utiliza DESCEND() em uma expressão INDEX para criar um índice de datas de  
// ordem descendente:
```

```
USE Sales NEW  
INDEX ON DESCEND(DTOS(OrdDate)) TO SalesDate
```

```
// Depois, DESCEND() pode ser utilizado para fazer uma pesquisa (SEEK) no índice  
// descendente:
```

```
DbSEEK(DESCEND(DTOS(dFindDate)))
```

---

## **GETDTOVAL()**

Função utilizada para retornar um numero formatado, de acordo com o valor passado por parâmetro, sendo que irá apenas manter os valores numéricos contidos na string passada por parâmetro, verificando se existe algum caractere '.' retornando um numero fracionário, na ordem dos números contidos na string.

A função é muito útil quando desejamos utilizar o valor numérico de uma data que está contida em uma string.

- Sintaxe:** GETDTOVAL ( < cDtoVal > )

- Parâmetros:**

<b>cDtoVal</b>	Representa uma string contendo um valor numérico no qual será convertido.
----------------	---

- Retorno:**

<b>Numérico</b>	Retorna um dado numérico de acordo com o valor informado em <cDtoVal>.
-----------------	--

**Exemplo:**

```
GetDtoVal('123456')      //retorno 123456.0000
GetDtoVal('1/2/3/4/5/6')  //retorno 123456.0000
GetDtoVal('fim.123456')  //retorno 0.123456
GetDtoVal('teste')       //retorno 0.0
```

**ISALPHA()**

Função utilizada para determinar se o caractere mais à esquerda em uma cadeia de caracteres é alfabético, permitindo avaliar se o string especificado começa com um caractere alfabético. Um caractere alfabético consiste em qualquer letra maiúscula ou minúscula de "A" a "Z".

- Sintaxe: ISALPHA ( < cString > )**

- Parâmetros:**

<b>cString</b>	Cadeia de caracteres a ser examinada.
----------------	---------------------------------------

- Retorno:**

<b>Lógico</b>	Retorna verdadeiro (.T.) se o primeiro caractere em <cString> for alfabético, caso contrário, retorna falso (.F.).
---------------	--

**ISDIGIT()**

Função utilizada para determinar se o caractere mais à esquerda em uma cadeia de caracteres é um dígito, permitindo avaliar se o primeiro caractere em um string é um dígito numérico entre zero e nove.

- Sintaxe: ISDIGIT ( < cString > )**

- Parâmetros:**

<b>cString</b>	Cadeia de caracteres a ser examinada.
----------------	---------------------------------------

- Retorno:**

<b>Lógico</b>	Retorna verdadeiro (.T.) caso o primeiro caractere da cadeia seja um dígito entre zero e nove; caso contrário, retorna falso (.F.).
---------------	---

## **ISLOWER()**

---

Função utilizada para determinar se o caractere mais à esquerda é uma letra minúscula, permitindo avaliar se o primeiro caractere de um string é uma letra minúscula. É o contrário de ISUPPER(), a qual determina se a cadeia de caracteres começa com uma letra maiúscula. ISLOWER() e ISUPPER() ambas são relacionadas às funções LOWER() e UPPER(), que convertem caracteres minúsculos para maiúsculos, e vice-versa.

- Sintaxe:** **ISLOWER( < cString > )**

- Parâmetros:**

<b>cString</b>	Cadeia de caracteres a ser examinada.
----------------	---------------------------------------

- Retorno:**

<b>Lógico</b>	Retorna verdadeiro (.T.) caso o primeiro caractere da cadeia seja minúsculo , caso contrário, retorna falso (.F.).
---------------	--

## **ISUPPER()**

---

Função utilizada para determinar se o caractere mais à esquerda é uma letra maiúscula, permitindo avaliar se o primeiro caractere de um string é uma letra maiúscula. É o contrário de ISLOWER (), a qual determina se a cadeia de caracteres começa com uma letra minúscula. ISLOWER() e ISUPPER() ambas são relacionadas às funções LOWER() e UPPER(), que convertem caracteres minúsculos para maiúsculos, e vice-versa.

- Sintaxe:** **ISUPPER( < cString > )**

- Parâmetros:**

<b>cString</b>	Cadeia de caracteres a ser examinada.
----------------	---------------------------------------

- Retorno:**

<b>Lógico</b>	Retorna verdadeiro (.T.) caso o primeiro caractere da cadeia seja maiúsculo , caso contrário, retorna falso (.F.).
---------------	--

## **LEN()**

---

Retorna o tamanho da string especificada no parâmetro.

- Sintaxe:** **LEN(cString)**

- Parâmetros**

<b>cString</b>	String que será avaliada
----------------	--------------------------

### **Exemplo:**

```
cNome := ALLTRIM(SA1->A1_NOME)
MSGINFO("Dados do campo A1_NOME:" + CRLF
" Tamanho:" + CVALTOCHAR(LEN(SA1->A1_NOME))+CRLF
"Texto:" + CVALTOCHAR(LEN(cNome)))
```

## **LOWER()**

---

Retorna uma string com todos os caracteres minúsculos, tendo como base a string passada como parâmetro.

- Sintaxe: LOWER(cString)**
- Parâmetros**

<b>cString</b>	String que será convertida para caracteres minúsculos.
----------------	--

### **Exemplo:**

```
cTexto := "ADVPL"  
MSGINFO("Texto:"+LOWER(cTexto))
```

## **LTRIM()**

---

Função para tratamento de caracteres utilizada para formatar cadeias de caracteres que possuam espaços em branco à esquerda. Pode ser o caso de, por exemplo, números convertidos para cadeias de caracteres através da função STR().

LTRIM() é relacionada a RTRIM(), a qual remove espaços em branco à direita, e a ALLTRIM(), que remove espaços tanto à esquerda quanto à direita.

O contrário de ALLTRIM(), LTRIM(), e RTRIM() são as funções PADC(), PADR(), e PADL(), as quais centralizam, alinham à direita, ou alinham à esquerda as cadeias de caracteres, através da inserção de caracteres de preenchimento.

- Sintaxe: LTRIM ( < cString > )**
- Parâmetros:**

<b>cString</b>	<cString> é a cadeia de caracteres a ser copiada sem os espaços em branco à esquerda.
----------------	---

### **Retorno:**

<b>Caracter</b>	LTRIM() retorna uma cópia de <cString>, sendo que os espaços em branco à esquerda foram removidos. Caso <cString> seja uma cadeia de caracteres nula ("") ou toda composta de espaços em branco, LTRIM() retorna uma cadeia de caracteres nula ("").
-----------------	--

## **MATHC()**

---

Função utilizada para realizar operações matemáticas com strings que contém um valor numérico. MATHC() realiza algumas operações matemáticas como: Soma, Subtração, Divisão, Multiplicação e Exponenciação.

A função irá retornar uma string contendo o resultado da operação matemática, com uma especificação de até 18 casas de precisão no numero.

- Sintaxe:** **MATHC ( < cNum1 > , < cOperacao > , < cNum2 > )**
- Parâmetros:**

<b>cNum1</b>	String contendo um valor numérico, representando o numero no qual desejamos realizar uma operação.
<b>cOperacao</b>	Representa a string que indica a operação que desejamos realizar. Olhar na tabela para verificar quais valores devem ser informados aqui.
<b>cNum2</b>	String contendo um valor numérico, representando o numero no qual desejamos realizar uma operação.

- Retorno:**

<b>Caracter</b>	Retorna uma nova string contendo o resultado matemático da operação.
-----------------	--

## **OEMTOANSI()**

---

Função que transforma uma string no Formato OEM / MS-DOS Text para uma string ANSI Text ( formato do Windows ).

Quando utilizamos um programa baseado no MS-DOS para alimentar uma base de dados , os acentos e caracteres especiais são gravados como texto OEM . Para tornar possível a correta visualização destes dados em uma interface Windows , utilizamos a função OemToAnsi() para realizar a conversão.

Ao utilizarmos um programa baseado no Windows para alimentar uma base de dados , o texto é capturado no formato ANSI Text . Caso este texto seja utilizado para alimentar uma base de dados a ser acessada através de um programa MS-DOS , devemos converter o dado para OEM antes de gravá-lo , através da função AnsiToOem().

- Sintaxe:** **OemToAnsi ( < cStringOEM > )**
- Parâmetros:**

<b>cStringOEM</b>	String em formato OEM - MsDos a ser convertida.
-------------------	---

- Retorno:**

<b>Caracter</b>	String convertida para ser exibida no Windows ( Formato ANSI ).
-----------------	---

## **PADL() / PADR() / PADC()**

---

Funções de tratamento de strings que inserem caracteres de preenchimento para completar um tamanho previamente especificado em vários formatos como data ou numéricos.

- PADC() centraliza <cExp>, adicionando caracteres de preenchimento à direita e à esquerda.
- PADL() adiciona caracteres de preenchimento à esquerda.
- PADR() adiciona caracteres de preenchimento à direita.

Caso o tamanho de <cExp> exceda o argumento <nTamanho>, todas as funções PAD() truncam string preenchida ao <nTamanho> especificado.

PADC(), PADL(), e PADR() são utilizadas para exibir cadeias de caracteres de tamanho variável em uma área de tamanho fixo. Elas podem ser usadas, por exemplo, para assegurar o alinhamento com comandos ?? consecutivos. Outra utilização é exibir textos em uma tela de tamanho fixo, para certificar-se de que o texto anterior foi completamente sobreescrito.

PADC(), PADL(), e PADR() são o contrário das funções ALLTRIM(), LTRIM(), e RTRIM(), as quais eliminam espaços em branco à esquerda e à direita de cadeias de caracteres.

- Sintaxe: PADL / PADR / PADC ( < cExp > , < nTamanho > , [ cCaracPreench ] )**
- Parâmetros**

<b>cExp</b>	Caractere, data, ou numérico no qual serão inseridos caracteres de preenchimento.
<b>nTamanho</b>	Tamanho da cadeia de caracteres a ser retornada.
<b>cCaracPreench</b>	Caractere a ser inserido em cExp. Caso não seja especificado, o padrão é o espaço em branco.

- Retorno:**

<b>Caracter</b>	Retorna o resultado de <cExp> na forma de uma cadeia de caracteres preenchida com <cCaracPreench>, para totalizar o tamanho especificado por <nTamanho>.
-----------------	--

## **RAT()**

---

Retorna a última posição de um caractere ou string dentro de outra string especificada.

- Sintaxe: RAT(cCaractere, cString)**
- Parâmetros**

<b>cCaractere</b>	Caractere ou string que se deseja verificar
<b>cString</b>	String na qual será verificada a existência do conteúdo de cCaractere.

## **REPLICATE()**

---

A função Replicate() é utilizada para gerar uma cadeira de caracteres repetidos a partir de um caracter base informado, podendo a string gerada conter até 64KB. Caso seja especificado no parâmetro de itens a repetir o número zero, será retornada uma string vazia.

- Sintaxe:** **REPLICATE(cString, nCount)**

- Parâmetros:**

<b>cString</b>	Caracter que será repetido
<b>nCount</b>	Quantidade de ocorrências do caracter base que serão geradas na string de destino.

- Retorno:**

<b>cReplicated</b>	String contendo as ocorrências de repetição geradas para o caracter informado.
--------------------	--

## **RETASC()**

---

A função Replicate() é utilizada para converter uma string numérica em uma informação composta por letras e números, devido a limitação no tamanho de campos ou variáveis de controle, como é o caso do campo de sistema X3\_ORDEM.

- Sintaxe:** **RETASC(cString, nTamanho, lVolta)**

- Parâmetros:**

<b>cString</b>	String a ser convertida
<b>nTamanho</b>	Tamanho máximo da string de retorno
<b>lVolta</b>	Indica se será gerada uma string composta por letras e números a partir de uma string apenas numérica (.T.) ou se será gerada uma string numérica a partir de uma string composta por letras e números (.F.)

- Retorno:**

<b>Caracter</b>	String convertida conforme o parâmetro lVolta.
-----------------	--

### **Exemplo1 – Conversão de numérico para alfanumérico**

```
cOrdem := "100"  
cX3Ordem := RETASC(cOrdem, 2, .T.) <-- Será retornado "A0"
```

### **Exemplo2 – Conversão de alfanumérico para numérico**

```
cX3Ordem:= "A0"  
cOrdem:= RETASC(cX3Ordem, 3, .F.) <-- Será retornado "100"
```

## RTRIM()

---

Função para tratamento de caracteres utilizada para formatar cadeias de caracteres que contenham espaços em branco à direita. Ela é útil quando você deseja eliminar espaços em branco à direita ao se concatenar cadeias de caracteres. É o caso típico com campos de banco de dados que são armazenados em formato de tamanho fixo. Por exemplo, você pode usar RTRIM() para concatenar o primeiro e o último campos de nome para formar uma cadeia de caracteres de nome.

LTRIM() é relacionada a RTRIM(), que remove espaços em branco à direita, e a ALLTRIM(), que remove espaços em branco à direita e à esquerda.

O contrário de ALLTRIM(), LTRIM(), e RTRIM() são as funções PADC(), PADR(), e PADL(), as quais centralizam, alinham à direita, ou alinham à esquerda cadeias de caracteres, inserindo caracteres de preenchimento.

- Sintaxe:** RTRIM ( < cString > ) --> cTrimString

**Parâmetros:**

<b>cString</b>	<cString> é a cadeia de caracteres a ser copiada sem os espaços em branco à direita.
----------------	--

**Retorno:**

<b>Caracter</b>	RTRIM() retorna uma cópia de <cString>, sendo que os espaços em branco à direita foram removidos. Caso <cString> seja uma cadeia de caracteres nula ("") ou totalmente composta por espaços, RTRIM() retorna uma cadeia de caracteres nula ("").
-----------------	--

## SPACE()

---

Função de tratamento de caracteres utilizada para retornar uma quantidade especificada de espaços. A utilização desta função tem o mesmo efeito que REPLICATE(' ', <nCont>), e é normalmente utilizada para inicializar uma variável do tipo caractere, antes que a mesma seja associada a um GET.

- Sintaxe:** SPACE ( < nCont > )

**Parâmetros:**

<b>nCont</b>	A quantidade de espaços a serem retornados, sendo que o número máximo é 65.535 (64K).
--------------	---

**Retorno:**

<b>Caracter</b>	Retorna uma cadeia de caracteres. Se <nCont> for zero, SPACE() retorna uma cadeia de caracteres nula ("").
-----------------	--

## **STRTOKARR()**

Função utilizada para retornar um array, de acordo com os dados passados como parâmetro para a função. Esta função recebe uma string <cValue> e um caracter <cToken> que representa um separador, e para toda ocorrência deste separador em <cValue> é adicionado um item no array.

- Sintaxe:** **STRTOKARR** ( < cValue > , < cToken > )

- Parâmetros:**

<b>cValue</b>	Representa a cadeia de caracteres no qual desejamos separar de acordo com <cToken>.
<b>cToken</b>	Representa o caracter que indica o separador em <cValue>.

- Retorno:**

<b>Array</b>	Array de caracteres que representa a string passada como parâmetro.
--------------	---

### **Exemplo:**

```
STRTOKARR('1;2;3;4;5', ';')      //retorna {'1','2','3','4','5'}
```

## **STRTRAN()**

Função utilizada para realizar a busca da ocorrência da string, sendo case sensitive.

- Sintaxe:** **STRTRAN** ( < cString > , < cSearch > , [ cReplace ] , [ nStart ] , [ nCount ] )

- Parâmetros:**

<b>cString</b>	Seqüência de caracteres ou campo memo a ser pesquisado.
<b>cSearch</b>	Seqüência de caracteres a ser procurada em cString.
<b>cReplace</b>	Seqüência de caracteres que deve substituir a string cSearch. Caso não seja especificado, as ocorrências de cSearch em cString serão substituídas por uma string nula ("").
<b>nStart</b>	nStart corresponde ao número seqüencial da primeira ocorrência de cSearch em cString a ser substituída por cReplace. Se este argumento for omitido, o default é 1 ( um ). Caso seja passado um numero menor que 1, a função retornará uma string em branco ("").
<b>nCount</b>	nCount corresponde ao número máximo de trocas que deverá ser realizada pela função . Caso este argumento não seja especificado , o default é substituir todas as ocorrências encontradas.

- Retorno:**

<b>Code-Block</b>	A função STRTRAN retorna uma nova string, com as ocorrências especificadas de cSearch trocadas para cReplace, conforme parametrização.
-------------------	--

## **STUFF()**

Função que permite substituir um conteúdo caractere em uma string já existente, especificando a posição inicial para esta adição e o número de caracteres que serão substituídos.

- Sintaxe: STUFF(cString, nPosInicial, nExcluir, cAdicao)**

- Parâmetros:**

<b>cString</b>	A cadeia de caracteres destino na qual serão eliminados e inseridos caracteres.
<b>nPosInicial</b>	A posição inicial na cadeia de caracteres destino onde ocorre a inserção/exclusão.
<b>nExcluir</b>	A quantidade de caracteres a serem eliminados.
<b>cAdicao</b>	A cadeia de caracteres a ser inserida.

- Retorno:**

<b>Caracter</b>	Retorna a nova string gerada pela função com as modificações.
-----------------	---

### **Exemplo:**

```
cLin := Space(100)+cEOL // Cria a string base
cCpo := PADR(SA1->A1_FILIAL,02) // Informação que será armazenada na string
cLin := Stuff(cLin,01,02,cCpo) // Substitui o conteúdo de cCpo na string base
```

## **SUBSTR()**

Retorna parte do conteúdo de uma string especificada, de acordo com a posição inicial deste conteúdo na string e a quantidade de caracteres que deverá ser retornada a partir daquele ponto (inclusive).

- Sintaxe: SUBSTR(cString, nPosInicial, nCaracteres)**
- Parâmetros**

<b>cString</b>	String que se deseja verificar
<b>nPosInicial</b>	Posição inicial da informação que será extraída da string
<b>nCaracteres</b>	Quantidade de caracteres que deverá ser retornada a partir daquele ponto (inclusive).

### **Exemplo:**

```
cCampo := "A1_NOME"
nPosUnder := AT(cCampo)
cPrefixo := SUBSTR(cCampo,1, nPosUnder) // → "A1_"
```

## **TRANSFORM()**

---

Função de conversão que formata valores caractere, data, lógicos e numéricos conforme um string de máscara especificado, a qual inclui uma combinação de strings de template e funções de picture. Ela faz o mesmo que a cláusula PICTURE do comando @...SAY, sendo normalmente utilizada para formatar dados a serem enviados à tela ou à impressora.

- Sintaxe:** **TRANSFORM ( < cExp > , < cSayPicture > )**

**Parâmetros:**

<b>cExp</b>	O valor a ser formatado. Esta expressão pode ser qualquer tipo de dados válidos, exceto vetor, bloco de código, e NIL.
<b>cSayPicture</b>	Uma string de caracteres de máscara e template usado para descrever o formato da cadeia de caracteres a ser retornada.

**Retorno:**

-	Retorna a conversão de <cExp> para uma cadeia de caracteres formatada conforme a definição em <cSayPicture>.
---	--

## **UPPER()**

---

Retorna uma string com todos os caracteres maiúsculos, tendo como base a string passada como parâmetro.

- Sintaxe:** **UPPER(cString)**  
 **Parâmetros**

<b>cString</b>	String que será convertida para caracteres maiúsculos.
----------------	--

**Exemplo:**

```
cTexto := "ADVPL"  
MSGINFO("Texto:"+LOWER(cTexto))
```



*Anotações*

---

---

---

---

---

## Manipulação de data / hora

### CDOW()

Função que converte uma data para uma cadeia de caracteres.

- Sintaxe:** CDOW( *dExp* )
- Parâmetros:**

<b>dExp</b>	Data que será convertida.
-------------	---------------------------

- Retorno:**

<b>cDayWeek</b>	Nome do dia da semana como uma cadeia de caracteres. A primeira letra é maiúscula e as demais minúsculas.
-----------------	---

#### Exemplo:

```
dData := DATE() // Resultado: 09/01/90  
cDiaDaSemana := CDOW(DATE()) // Resultado: Friday  
cDiaDaSemana := CDOW(DATE() + 7) // Resultado: Friday  
cDiaDaSemana := CDOW(CTOD("06/12/90")) // Resultado: Tuesday
```



Importante

A função **FG\_CDOW(dExp)** retorna o nome do dia da semana de acordo com o idioma em uso pelo ERP.

### CMONTH()

Função de conversão de datas que retorna uma cadeia de caracteres com o nome do mês em inglês.

- Sintaxe:** CMONTH( *dData* )
- Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

- Retorno:**

<b>cMonth</b>	Retorna o nome do mês em uma cadeia de caracteres. A primeira letra do retorno em maiúscula e o restante do nome, em minúsculas.
---------------	--

#### Exemplo:

```
cMes := CMONTH(DATE()) // Resultado: September  
cMes := CMONTH(DATE() + 45) // Resultado: October  
cMes := CMONTH(CTOD("12/01/94")) // Resultado: December  
cMes := SUBSTR(CMONTH(DATE()), 1, 3) + STR(DAY(DATE())) // Resultado: Sep 1
```

## **DATE()**

Função que retorna a data do atual sistema. O formato de saída é controlado pelo comando SET DATE, sendo que o formato padrão é mm/dd/yy.

- Sintaxe: DATE()**
- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>dData</b>	Data do sistema.
--------------	------------------

### **Exemplo:**

```
dData := DATE() // Resultado: 09/01/01  
dData := DATE() + 30 // Resultado: 10/01/01  
dData := DATE() - 30 // Resultado: 08/02/90  
dData := DATE()  
cMes := CMONTH(dData) // Resultado: September
```

## **DAY()**

Função de conversão de datas usada para converter o valor data em um número inteiro que representa o dia do mês. Esta função pode ser usada em conjunto com CMONTH() e YEAR() para formatar datas. Pode ser usada também em diversos cálculos envolvendo datas.

- Sintaxe: DAY( *dData* )**
- Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

- Retorno:**

<b>nDias</b>	Se o mês do argumento dData for fevereiro, anos bissextos são considerados. Se a data do argumento dData for 29 de fevereiro e o ano não for bissexto, ou se o argumento dData for vazio.
--------------	---

### **Exemplo:**

#### **// Estes exemplos mostram a função DAY() de diversas maneiras:**

```
dData := DATE() // Resultado: 09/01/01
```

```
nDia := DAY(DATE()) // Resultado: 1
```

```
nDia := DAY(DATE()) + 1 // Resultado: 2
```

```
nDia := DAY(CTOD("12/01/94")) // Resultado: 1
```

#### **// Este exemplo mostra a função DAY() usada em conjunto com CMONTH() e**

YEAR() para formatar o valor da data:

```
dData := Date()
```

```
cData := CMONTH(dData) + STR(DAY(dData)) + "," + STR(YEAR(dData)) // Resultado: June  
15, 2001
```

## **DOW()**

Função que converte uma data para o valor numérico que representa o dia da semana. Útil quando se deseja fazer cálculos semanais. DOW() é similar a CDOW(), que retorna o dia da semana como uma cadeia de caracteres.

- Sintaxe: DOW( *dData* )**
- Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

- Retorno:**

<b>nDia</b>	Retorna um número entre zero e sete, representando o dia da semana. O primeiro dia da semana é 1 (Domingo) e o último é 7 (Sábado). Se a data for vazia ou inválida, DOW() retorna zero.
-------------	--

### **Exemplo:**

```
dData := DATE() // Resultado: 09/01/01  
nDiaDaSemana := DOW(DATE()) // Resultado: 3  
cDiaDaSemana := CDOW(DATE()) // Resultado: Tuesday  
nDiaDaSemana := DOW(DATE() - 2) // Resultado: 1  
cDiaDaSemana := CDOW(DATE() - 2) // Resultado: Sunday
```

## **DTOC()**

Função para conversão de uma data para uma cadeia de caracteres formatada segundo o padrão corrente, definido pelo comando SET DATE. Se for necessária a utilização de formatação especial, use a função TRANSFORM().

Em expressões de índices de arquivo, use DTOS() no lugar de DTOC() para converter datas para cadeia de caracteres.

- Sintaxe: DTOC( *dData* )**
- Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

- Retorno:**

<b>cData</b>	É uma cadeia de caracteres representando o valor da data. O retorno é formatado utilizando-se o formato corrente definido pelo comando SET DATE FORMAT. O formato padrão é mm/dd/yy. Para uma data nula ou inválida, o retorno será uma cadeia de caracteres com espaços e tamanho igual ao formato atual.
--------------	--

### **Exemplo:**

```
cData := DATE() // Resultado: 09/01/90  
cData := DTOC(DATE()) // Resultado: 09/01/90  
cData := "Today is " + DTOC(DATE()) // Resultado: Today is 09/01/90
```

## **DTOS()**

Função para conversão de uma data que pode ser usada para criar expressões de índice. O resultado é estruturado visando manter a ordem correta do índice (ano, mês, dia).

- Sintaxe:** **DTOS( *dData* )**
- Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

- Retorno:**

<b>sData</b>	Retorna uma cadeia de caracteres com oito byte de tamanho no formato yyyyymmdd. Quando <i>dData</i> é nulo ou invalido, DTOS() retorna uma cadeia de caracteres com oito espaços. O valor retornado não é afetado pela formato da data corrente.
--------------	--

### **Exemplo:**

```
cData := DATE() // Resultado: 09/01/90  
cData := DTOS(DATE()) // Resultado: 19900901  
nLen := LEN(DTOS(CTOD(""))) // Resultado: 8
```

## **ELAPTIME()**

Função que retorna uma cadeia de caracteres contendo a diferença de tempo no formato hh:mm:ss, onde hh é a hora ( 1 a 24 ), mm os minutos e ss os segundos.

- Sintaxe:** **ElapTime( *cHoraInicial* , *cHoraFinal* )**
- Parâmetros:**

<b>cHoraInicial</b>	Informe a hora inicial no formato hh:mm:ss, onde hh é a hora ( 1 a 24 ), mm os minutos e ss os segundos
<b>cHoraFinal</b>	Informe a hora final no formato hh:mm:ss, onde hh é a hora ( 1 a 24 ), mm os minutos e ss os segundos.

- Retorno:**

<b>Caracter</b>	A diferença de tempo no formato hh:mm:ss, onde hh é a hora ( 1 a 24 ), mm os minutos e ss os segundos.
-----------------	--

### **Exemplo:**

```
cHoraInicio := TIME() // Resultado: 10:00:00  
...  
<instruções>  
...  
cElapsed := ELAPTIME(TIME(), cHoraInicio)
```

## **MONTH()**

Função de conversão que extrai da data o valor numérico do mês, semelhante a função que retorna o nome do mês a partir do valor de dData.

- Sintaxe: MONTH( *dData* )**
- Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

- Retorno:**

<b>Numérico</b>	>=0 e <=12 → Para uma data válida. 0 → Se a data for nula ou inválida
-----------------	--

## **Exemplo:**

```
dData := DATE() // Resultado: 09/01/01  
nMes := MONTH(DATE()) // Resultado: 9  
nMes := MONTH(DATE()) + 1 // Resultado: 10
```

## **SECONDS()**

Esta função retorna o número de segundos decorridos desde a meia-noite, segundo a hora do sistema. Está relacionada à função TIME() que retorna a hora do sistema como uma cadeia de caracteres no formato hh:mm:ss.

- Sintaxe: SECONDS()**
- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Numérico</b>	>=0 e <=86399 → Retorna a hora do sistema em segundos. O valor numérico representa o número de segundos decorridos desde a meia-noite, baseado no relógio de 24 horas e varia de 0 a 86399.
-----------------	---

## **Exemplo:**

```
cHora := TIME() // Resultado: 10:00:00  
cSegundos := SECONDS() // Resultado: 36000.00
```

//Este exemplo usa a função SECONDS() para cronometrar o tempo decorrido:

```
LOCAL nStart, nElapsed  
nStart:= SECONDS()
```

## **TIME()**

Função que retorna a hora do sistema como uma cadeia de caracteres, e que está relacionada com SECONDS(), que retorna o valor inteiro representando o número de segundos desde a meia-noite. SECONDS() é geralmente usada no lugar de TIME() para cálculos.

**Sintaxe:** TIME()

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Caracter</b>	A hora do sistema como uma cadeia de caracteres no formato hh:mm:ss onde hh é a hora ( 1 a 24 ), mm os minutos e ss os segundos.
-----------------	--

**Exemplo:**

```
cTime := TIME() // Resultado: 10:37:17  
cHora := SUBSTR(cTime, 1, 2) // Resultado: 10  
cMinutos := SUBSTR(cTime, 4, 2) // Resultado: 37  
cSegundos := SUBSTR(cTime, 7, 2) // Resultado: 17
```

## **YEAR()**

YEAR() é uma função de conversão de data que extrai o valor numérico do ano. YEAR() é membro de um grupo de funções que retornam valores numéricos de uma data. O grupo inclui DAY() e MONTH() que retornam o dia e o mês como valores numéricos.

**Sintaxe:** YEAR( dData )

**Parâmetros:**

<b>dData</b>	Data que será convertida.
--------------	---------------------------

**Retorno:**

<b>Numérico</b>	Valor numérico do ano da data especificada em dData incluindo os dígitos do século. O valor retornado não é afetado pelos valores especificados pelos comandos SET DATE ou SET CENTURY.
-----------------	---

**Exemplo 01:**

```
dData := DATE() // Resultado: 09/20/01  
dAno := YEAR(dData) // Resultado: 2001  
dAno := YEAR(dData) + 11 // Resultado: 2012
```

### **Exemplo 02:**

```
// Este exemplo cria uma função de usuário que usa a função YEAR() para formatar o valor da  
// data:
```

```
cData := Mdy(DATE()) // Result: September 20, 1990  
FUNCTION Mdy( dDate )  
RETURN CMONTH(dDate) + " " + LTRIM(STR(DAY(dDate))) + "," + STR(YEAR(dDate))
```

## **Manipulação de variáveis numéricas**

### **ABS()**

Retorna um valor absoluto (independente do sinal) com base no valor especificado no parâmetro.

- Sintaxe: ABS(nValor)**
- Parâmetros**

<b>nValor</b>	Valor que será avaliado
---------------	-------------------------

### **Exemplo:**

```
nPessoas := 20  
nLugares := 18  
  
IF nPessoas < nLugares  
    MSGINFO("Existem "+CVALTOCHAR(nLugares- nPessoas)+"disponíveis")  
ELSE  
    MSGSTOP("Existem "+CVALTOCHAR(ABS(nLugares- nPessoas))+ "faltando")  
ENDIF
```

### **ALEATORIO()**

Gera um número aleatório de acordo com a semente passada. Esta função retorna um número aleatório menor ou igual ao primeiro parâmetro informado, usando como semente o segundo parâmetro. É recomendado que esta semente seja sempre o último número aleatório gerado por esta função.

- Sintaxe: Aleatorio(nMax,nSeed)**
- Parâmetros**

<b>nMax</b>	Número máximo para a geração do número aleatório
<b>nSeed</b>	Semente para a geração do número aleatório

### **Exemplo – Função ALEATORIO()**

```
nSeed := 0
For i := 1 to 100
nSeed := Aleatorio(100,nSeed)
? Str(i,3)+"§ numero aleatorio gerado: "+Str(nSeed,3)
Next i
inkey(0)
Return
```

### **INT()**

Retorna a parte inteira de um valor especificado no parâmetro.

- Sintaxe: INT(nValor)**
- Parâmetros**

<b>nValor</b>	Valor que será avaliado
---------------	-------------------------

### **Exemplo:**

```
STATIC FUNCTION COMPRAR(nQuantidade)
```

```
LOCAL nDinheiro := 0.30
LOCAL nPrcUnit := 0.25

IF nDinheiro >= (nQuantidade*nPrcUnit)
    RETURN nQuantidade
ELSEIF nDinheiro > nPrcUnit
    nQuantidade := INT(nDinheiro / nPrcUnit)
ELSE
    nQuantidade := 0
ENDIF

RETURN nQuantidade
```

### **NOROUND()**

Retorna um valor, truncando a parte decimal do valor especificado no parâmetro de acordo com a quantidade de casas decimais solicitadas.

- Sintaxe: NOROUND(nValor, nCasas)**
- Parâmetros**

<b>nValor</b>	Valor que será avaliado
<b>nCasas</b>	Número de casas decimais válidas. A partir da casa decimal especificada os valores serão desconsiderados.

### **Exemplo – Função NOROUND()**

```
nBase := 2.985
nValor := NOROUND(nBase,2) → 2.98
```

### **RANDOMIZE()**

Através da função RANDOMIZE() , geramos um numero inteiro aleatório, compreendido entre a faixa inferior e superior recebida através dos parâmetros nMinimo e nMaximo, respectivamente.

#### **Observação :**

- O limite inferior recebido através do parâmetro nMinimo é "maior ou igual a ", podendo ser sorteado e fazer parte do retorno; porém o limite superior é "menor que", de modo a nunca será atingido ou devolvido no resultado. Por exemplo , a chamada da função RANDOMIZE(1,2) sempre retornará 1 .
- Sintaxe: RANDOMIZE ( < nMinimo > , < nMaximo > )**
- Parâmetros**

<b>nMinimo</b>	Corresponde ao menor numero a ser gerado pela função.
<b>nMaximo</b>	Corresponde ao maior número ( menos um ) a ser gerado pela função.

#### **Retorno:**

<b>Numérico</b>	Numero randômico , compreendido no intervalo entre (nMinimo) e (nMaximo-1) : O numero gerado pode ser maior ou igual à nMinimo e menor ou igual a nMaximo-1 .
-----------------	---

### **ROUND()**

Retorna um valor, arredondando a parte decimal do valor especificado no parâmetro de acordo com a quantidades de casas decimais solicitadas, utilizando o critério matemático.

- Sintaxe: ROUND(nValor, nCasas)**
- Parâmetros**

<b>nValor</b>	Valor que será avaliado
<b>nCasas</b>	Número de casas decimais válidas. As demais casas decimais sofrerão o arredondamento matemático, aonde:  Se $nX \leq 4 \rightarrow 0$ , senão +1 para a casa decimal superior.

#### **Exemplo:**

```
nBase := 2.985
nValor := ROUND(nBase,2) → 2.99
```

## Manipulação de arquivos

### ADIR()

Função que preenche os arrays passados com os dados dos arquivos encontrados, através da máscara informada. Tanto arquivos locais (Remote) como do servidor podem ser informados.

**Importante:** ADir é uma função obsoleta, utilize sempre Directory().

- Sintaxe:** ADIR([ **cArqEspec** ], [ **aNomeArq** ], [ **aTamanho** ], [ **aData** ], [ **aHora** ], [ **aAtributo** ])

**Parâmetros:**

<b>cArqEspec</b>	Caminho dos arquivos a serem incluídos na busca de informações. Segue o padrão para especificação de arquivos, aceitando arquivos no servidor Protheus e no Cliente. Caracteres como * e ? são aceitos normalmente. Caso seja omitido, serão aceitos todos os arquivos do diretório default ( *.* ).
<b>aNomeArq</b>	Array de Caracteres. É o array com os nomes dos arquivos encontrados na busca. O conteúdo anterior do array é apagado.
<b>aTamanho</b>	Array Numérico. São os tamanhos dos arquivos encontrados na busca.
<b>aData</b>	Array de Datas. São as datas de modificação dos arquivos encontrados na busca.
<b>aHora</b>	Array de Caracteres. São os horários de modificação dos arquivos encontrados. Cada elemento contém horário no formato: hh:mm:ss.
<b>aAtributos</b>	Array de Caracteres. São os atributos dos arquivos, caso esse array seja passado como parâmetros, serão incluídos os arquivos com atributos de sistema e ocultos.

**Retorno:**

<b>nArquivos</b>	Quantidade de arquivos encontrados.
------------------	-------------------------------------

**Exemplo:**

```
LOCAL aFiles[ADIR("*.TXT")]
ADIR("*.TXT", aFiles)
AEVAL(aFiles, { |element| QOUT(element) })
```



Anotações

---

---

---

---

## **CGETFILE()**

Função utilizada para seleção de um arquivo ou diretório, disponibilizando uma interface gráfica para amigável para o usuário. Esta função está normalmente associada ao recurso de abrir ou salvar arquivos, permitindo para esta última a digitação opcional do nome do arquivo que será gravado.

- Sintaxe:** **cGetFile ( ExpC1, ExpC2, ExpN1, ExpC3, ExpL1, ExpN2,ExpL2 )**

- Parâmetros:**

<b>ExpC1</b>	Mascara para filtro (Ex: 'Informes Protheus (*.# #R)   *.# #R')
<b>ExpC2</b>	Titulo da Janela
<b>ExpN1</b>	Numero da mascara default ( Ex: 1 p/ *.exe )
<b>ExpC3</b>	Diretório inicial se necessário
<b>Expl1</b>	.T. para mostrar botão como 'Salvar' e .F. para botão 'Abrir'
<b>ExpN2</b>	Mascara de bits para escolher as opções de visualização do Objeto.
<b>ExpL2</b>	.T. para exibir diretório [Servidor] e .F. para não exibir

- Máscaras de bits para opções:**

<b>GETF_OVERWRITEPROMPT</b>	Solicita confirmação para sobrescrever
<b>GETF_MULTISELECT</b>	Permite selecionar múltiplos arquivos
<b>GETF_NOCHANGEDIR</b>	Não permite mudar o diretório inicial
<b>GETF_LOCALFLOPPY</b>	Exibe o(s) Drive(s) de disquete da maquina local
<b>GETF_LOCALHARD</b>	Exibe o(s) HardDisk(s) Local(is)
<b>GETF_NETWORKDRIVE</b>	Exibe os drives da rede ( Mapeamentos )
<b>GETF_SHAREWARE</b>	Não implementado
<b>GETF_RETDIRECTORY</b>	Retorna um diretório

### **Exemplo:**

```
cGetFile ( '*.PRW|*.CH' , 'Fontes', 1, 'C:\VER507', .F., GETF_LOCALHARD +
GETF_LOCALFLOPPY)
```



*Anotações*

---



---



---

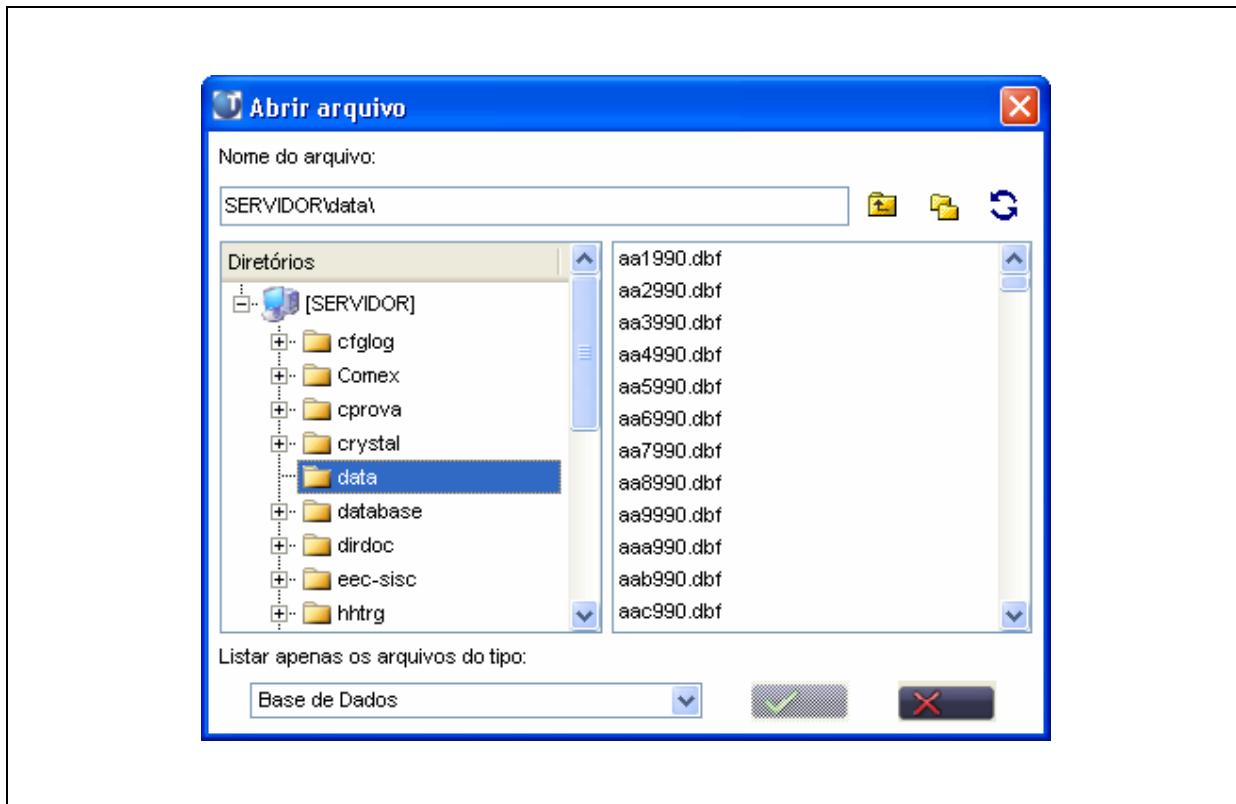


---



---

**Aparência:**



Para permitir a seleção de diversos arquivos contidos em um diretório é necessário combinar as funções CGETFILE(), DIRECTORY() e o objeto LISTBOX() conforme abaixo:



Dica

- CGETFILE: exibe os diretórios disponíveis e retorna o nome do item selecionado.
- DIRECTORY: efetua a leitura dos arquivos contidos no diretório retornado pela CGETFILE.
- LISTBOX: Exibe uma tela de seleção de com a opção de marcação, para que sejam selecionados os arquivos que serão processados.



Anotações

---

---

---

---

## **Exemplo: Seleção de múltiplos arquivos com CGETFILE, DIRECTORY() e LISTBOX()**

### **Função Principal: SELFILE()**

```
#include "protheus.ch"

//+-----+
//| Rotina | SELFILE | Autor | ARNALDO R. JUNIOR | Data | 01.01.2007 |
//+-----+
//| Descr. | Função exemplo para seleção de múltiplos arquivos. |
//+-----+
//| Uso     | CURSO DE ADVPL |
//+-----+

USER FUNCTION SELFILE()

LOCAL cDirectory := ""
LOCAL aArquivos    := {}
LOCAL nArq        := 0

PRIVATE aParamFile:= ARRAY(1)

IF !PARBOXFILE()
    RETURN
ENDIF

// Exibe a estrutura de diretório e permite a seleção dos arquivos que serão
processados
cDirectory := ALLTRIM(cGetFile("Arquivos de Dados|" + aParamFile[1] + " |",
'Importação de lançamentos', 0, '', .T., GETF_OVERWRITEPROMPT + GETF_NETWORKDRIVE
+ GETF_RETDIRECTORY,.T.))
aArquivos   := Directory(cDirectory+"*.*")
aArquivos   := MARKFILE(aArquivos,cDirectory,aParamFile[1],@lSelecao)

FOR nArq TO Len(aArquivos)

    IF !aArquivos[nArq][1]
        LOOP
    ENDIF

    <...processamento...>

NEXT nArq

RETURN
```

## Função auxiliar: PARBOXFILE()

```
//+-----+
//| Rotina | PARBOXFILE | Autor | ARNALDO R. JUNIOR Data | 01.01.2007 |
//+-----+
//| Descr. | Função exemplo de uso da PARAMBOX em conjunto com CGETFILE|
//+-----+
//| Uso     | CURSO DE ADVPL                         |
//+-----+



STATIC FUNCTION PARBOXFILE()

Local aParamBox := {}
Local cTitulo      := "Filtros Adicionais"
Local aRet       := {}
Local bOk        := {|| .T.}
Local aButtons    := {}
Local lCentered   := .T.
Local nPosx
Local nPosy
Local cLoad       := ""
Local lCanSave    := .F.
Local lUserSave   := .F.
Local nX          := 0
Local lRet        := .T.

AADD(aParamBox,{2,"Tipo de arquivo"
,2, {"*.dbf","*.dtc"},100,"AllwaysTrue()",.T.})

lRet := ParamBox(aParamBox, cTitulo, aRet, bOk, aButtons, lCentered, nPosx,
nPosy,, cLoad, lCanSave, lUserSave)

IF ValType(aRet) == "A" .AND. Len(aRet) == Len(aParamBox)
    For nX := 1 to Len(aParamBox)
        If aParamBox[nX][1] == 1
            aParam102[nX] := aRet[nX]
        ElseIf aParamBox[nX][1] == 2 .AND. ValType(aRet[nX]) == "C"
            aParam102[nX] := aRet[nX] // Tipo do arquivo
        ElseIf aParamBox[nX][1] == 2 .AND. ValType(aRet[nX]) == "N"
            aParam102[nX] := aParamBox[nX][4][aRet[nX]] // Tipo do arquivo
        Endif
    Next nX
ENDIF

RETURN lRet
```

## **Função auxiliar: MARKFILE()**

```
//+-----+
//| Rotina | MARKFILE | Autor | ARNALDO R. JUNIOR | Data | 01.01.2007 |
//+-----+
//| Descr. | Função exemplo para marcação de múltiplos arquivos. |
//+-----+
//| Uso     | CURSO DE ADVPL |
//+-----+

STATIC FUNCTION MARKFILE(aArquivos,cDiretorio,cDriver,lSelecao)

Local aChaveArq    := {}
Local cTitulo      := "Arquivos para importação: "
Local bCondicao   := {|| .T.}
// Variáveis utilizadas na seleção de categorias
Local oChkQual,lQual,oQual,cVarQ
// Carrega bitmaps
Local oOk          := LoadBitmap( GetResources(), "LBOK")
Local oNo          := LoadBitmap( GetResources(), "LBNO")
// Variáveis utilizadas para lista de filiais
Local nx          := 0
Local nAchou      := 0

//+-----+
//| Carrega os arquivos do diretório no array da ListBox |
//+-----+
For nX := 1 to Len(aArquivos)
    //+-----+
    //| aChaveArq - Contem os arquivos que serão exibidos para seleção |
    //+-----+
    AADD(aChaveArq, {.F.,aArquivos[nX][1],cDiretorio})
Next nx

//+-----+
//| Monta tela para seleção dos arquivos contidos no diretório |
//+-----+
DEFINE MSDIALOG oDlg TITLE cTitulo STYLE DS_MODALFRAME From 145,0 To 445,628;
OF oMainWnd PIXEL
oDlg:lEscClose := .F.
@ 05,15 TO 125,300 LABEL UPPER(cDriver) OF oDlg PIXEL
@ 15,20 CHECKBOX oChkQual VAR lQual PROMPT "Inverte Seleção" SIZE 50, 10;
OF oDlg PIXEL;
ON CLICK (AEval(aChaveArq, {|z| z[1] := If(z[1]==.T.,.F.,.T.)}),;
oQual:Refresh(.F.))
@ 30,20 LISTBOX oQual VAR cVarQ Fields HEADER "", "Código", "Descrição" SIZE;
273,090 ON DBLCLICK (aChaveArq:=Troca(oQual:nAt,aChaveArq),oQual:Refresh());
NoScroll OF oDlg PIXEL
oQual:SetArray(aChaveArq)
oQual:bLine := { || {If(aChaveArq[oQual:nAt,1],oOk,oNo),;
aChaveArq[oQual:nAt,2],aChaveArq[oQual:nAt,3]}}
DEFINE SBUTTON FROM 134,240 TYPE 1 ACTION IIF(MarcaOk(aChaveArq),;
(lSelecao := .T., oDlg:End(),.T.),.F.) ENABLE OF oDlg
DEFINE SBUTTON FROM 134,270 TYPE 2 ACTION (lSelecao := .F., oDlg:End());
ENABLE OF oDlg
ACTIVATE MSDIALOG oDlg CENTERED

RETURN aChaveArq
```

## **Função auxiliar: TROCA()**

```
//+-----+
//| Rotina | TROCA      | Autor | ARNALDO R. JUNIOR | Data | 01.01.2007 |
//+-----+
//| Uso     | CURSO DE ADVPL          |
//+-----+

STATIC FUNCTION Troca(nIt,aArray)
aArray[nIt,1] := !aArray[nIt,1]
Return aArray
```

## **Função auxiliar: MARCAOK()**

```
//+-----+
//| Rotina | MARCAOK    | Autor | ARNALDO R. JUNIOR | Data | 01.01.2007 |
//+-----+
//| Uso     | CURSO DE ADVPL          |
//+-----+

STATIC FUNCTION MarcaOk(aArray)
Local lRet:=.F.
Local nx:=0

// Checa marcações efetuadas
For nx:=1 To Len(aArray)
    If aArray[nx,1]
        lRet:=.T.
    EndIf
Next nx
// Checa se existe algum item marcado na confirmação
If !lRet
    HELP("SELFILE",1,"HELP","SEL. FILE","Não existem itens marcados",1,0)
EndIf

Return lRet
```



**Anotações**

---

---

---

---

## **CPYS2T()**

Função utilizada para copiar um arquivo do servidor para o cliente (Remote), sendo que os caracteres "\*" e "?" são aceitos normalmente. Caso a compactação seja habilitada (*ICompacta*), os dados serão transmitidos de maneira compacta e descompactados antes do uso.

**Sintaxe:** CPYS2T ( < cOrigem > , < cDestino > , [ ICompacta ] )

**Parâmetros:**

<b>cOrigem</b>	Nome(s) dos arquivos a serem copiados, aceita apenas arquivos no servidor, WildCards (* e ?) são aceitos normalmente.
<b>cDestino</b>	Diretório com o destino dos arquivos no Client ( Remote ).
<b>ICompacta</b>	Indica se a cópia deve ser feita compactando o arquivo antes do envio.

**Retorno:**

<b>Lógico</b>	ISucess retorna .T. caso o arquivo seja copiado com sucesso , ou .F. em caso de falha na cópia.
---------------	---

### **Exemplo:**

```
// Copia arquivos do servidor para o remote local, compactando antes de transmitir
CpyS2T( "\BKP\MANUAL.DOC", "C:\TEMP", .T. )
// Copia arquivos do servidor para o remote local, sem compactar antes de transmitir
CpyS2T( "\BKP\MANUAL.DOC", "C:\TEMP", .F. )
```

## **CPYT2S()**

Função utilizada para copiar um arquivo do cliente (Remote) para o servidor, sendo que os caracteres "\*" e "?" são aceitos normalmente. Caso a compactação seja habilitada (*ICompacta*), os dados serão transmitidos de maneira compacta e descompactados antes do uso.

**Sintaxe:** CpyT2S( cOrigem, cDestino, [ ICompacta ] )

**Parâmetros:**

<b>cOrigem</b>	Nomes dos arquivos a serem copiados, aceita apenas arquivos locais ( Cliente ), WildCards são aceitos normalmente.
<b>cDestino</b>	Diretório com o destino dos arquivos no remote ( Cliente ).
<b>ICompacta</b>	Indica se a cópia deve ser feita compactando o arquivo antes.

**Retorno:**

<b>Lógico</b>	Indica se o arquivo foi copiado para o cliente com sucesso.
---------------	---

### **Exemplo:**

```
// Copia arquivos do cliente( remote ) para o Servidor compactando antes de transmitir
CpyT2S( "C:\TEMP\MANUAL.DOC", "\BKP", .T. )
// Copia arquivos do cliente( remote ) para o Servidor sem compactar.
CpyT2S( "C:\TEMP\MANUAL.DOC", "\BKP" )
```

## **CURDIR()**

Função que retorna o diretório corrente do servidor. O caminho retornado é sempre relativo ao RootPath definido na configuração do Environment no .INI do Protheus Server. Inicialmente , o diretório atual da aplicação é o constante na chave StartPath , também definido na configuração do Environment no .INI do Protheus Server.

Caso seja passado o parâmetro cNovoPath , este path é assumido como sendo o Path atual. Caso o path recebido como parâmetro não exista , seja inválido , ou seja um path absoluto (iniciado com uma letra de drive ou caminho de rede), a função não irá setar o novo path, mantendo o atual .

- Sintaxe:** CURDIR ( [ cNovoPath ] )

- Parâmetros:**

<b>cNovoPath</b>	Caminho relativo , com o novo diretório que será ajustado como corrente.
------------------	--

- Retorno:**

<b>Caracter</b>	Diretório corrente, sem a primeira barra.
-----------------	---

### **Exemplo:**

```
cOldDir := curdir()
cNewDir := '\webadv\xis'
curdir(cNewDir) // Troca o path
If cNewDir <> '\'+curdir() // E verifica se trocou mesmo
    conout('Falha ao Trocar de Path de '+cOldDir + ' para '+cNewDir)
Else
    conout('Path de '+cOldDir + ' trocado para '+cNewDir+' com sucesso.')
Endif
```



*Anotações*

---

---

---

---

## **DIRECTORY()**

---

Função de tratamento de ambiente que retorna informações a respeito dos arquivos no diretório corrente ou especificado. É semelhante a ADIR(), porém retorna um único vetor ao invés de adicionar valores a uma série de vetores existentes passados por referência.

DIRECTORY() pode ser utilizada para realizar operações em conjuntos de arquivos. Em combinação com AEVAL(), você pode definir um bloco que pode ser aplicado a todos os arquivos que atendam a <cDirSpec> especificada.

Para tornar as referências aos vários elementos de cada sub-vetor de arquivo mais legíveis, é fornecido o arquivo header Directry.ch, que contém os #defines para os subarray subscripts.

**TABELA A: Atributos de DIRECTORY()**

Atributo	Significado
<b>H</b>	Incluir arquivos ocultos
<b>S</b>	Incluir arquivos de sistema
<b>D</b>	Incluir diretórios
<b>V</b>	Procura pelo volume DOS e exclui outros arquivos

*Nota: Arquivos normais são sempre incluídos na pesquisa, a não ser que V seja especificado.*

**TABELA B: Estrutura dos Subvetores de DIRECTORY()**

Posição	Metasímbolo	Directry.ch
<b>1</b>	cNome	F_NAME
<b>2</b>	cTamanho	F_SIZE
<b>3</b>	dData	F_DATE
<b>4</b>	cHora	F_TIME
<b>5</b>	cAtributos	F_ATT

**Sintaxe: DIRECTORY ( < cDirSpec > , [ ] )**

**Parâmetros:**

<b>cDirSpec</b>	<cDirSpec> especifica o drive, diretório e arquivo para a pesquisa no diretório. Caracteres do tipo coringa são permitidos na especificação de arquivos. Caso <cDirSpec> seja omitido, o valor padrão é *.*. O caminho especificado pode estar na estação (remote) , ou no servidor, obedecendo às definições de Path Absoluto / Relativo de acesso.
<b>cAtributos&gt;</b>	<cAtributos> especifica que arquivos com atributos especiais devem ser incluídos na informação retornada. <cAtributos> consiste em uma cadeia de caracteres que contém um ou mais dos seguintes caracteres, contidos na tabela adicional A , especificada anteriormente.

**Retorno:**

<b>Array</b>	DIRECTORY() retorna um vetor de sub-vetores, sendo que cada sub-vetor contém informações sobre cada arquivo que atenda a <cDirSpec>. Veja maiores detalhes na Tabela B, discriminada anteriormente.
--------------	---

**Exemplo:**

```
#INCLUDE "Directry.ch"

aDirectory := DIRECTORY(".*.*","D")
AEVAL( aDirectory, {|aFile| CONOUT(aFile[F_NAME])} )
```

## **DIRREMOVE()**

Função que elimina um diretório específico. Caso especifiquemos um path sem a unidade de disco , ele será considerado no ambiente do Servidor , a partir do RootPath do ambiente ( caso o path comece com \ ), ou a partir do diretório corrente (caso o path não seja iniciado com \ ).

Quando especificado um path absoluto ( com unidade de disco preenchida ), a função será executada na estação onde está sendo executado o Protheus Remote. Quando executamos a função DirRemove() em JOB ( processo isolado no Server , sem interface ), não é possível especificar um Path absoluto de disco. Caso isto seja realizado , a função retornará .F. e FError() retornará -1 ( Syntax Error ).

Note que é necessário ter direitos suficientes para remover um diretório, e o diretório a ser eliminado precisa estar vazio, sem subdiretórios ou arquivos dentro do mesmo.

**Sintaxe: DIRREMOVE ( < cDiretorio > )**

**Parâmetros:**

<b>cDiretorio</b>	Nome do diretório a ser removido.
-------------------	-----------------------------------

**Retorno:**

<b>Lógico</b>	ISucesso será .T. caso o diretório tenha sido eliminado , ou .F. caso não seja possível excluir o diretório. Quando a função DirRemove retornar .F. , é possível obter mais detalhes da ocorrência recuperando o código do Erro através da função FError().
---------------	---

**Exemplo:**

```
cDelPath := 'c:\TmpFiles'
IRemoveOk := DIRREMOVE(cDelPath)

IF !IRemoveOk
  MsgStop('Falha ao remover a pasta '+cDelPath+' ( File Error '+str(Fewrror(),4)+' ) ')
Else
  MsgStop('Pasta '+cDelPath+' removida com sucesso.')
Endif
```

## **DISKSPACE()**

Função de ambiente que determina quantos bytes estão disponíveis em uma determinada unidade de disco. Esta função obtém a informação sempre relativa à estação onde está sendo executado o Protheus Remote. Através do parâmetro nDrive , selecionamos qual a unidade de disco que desejamos obter a informação do espaço livre , onde:

- 0 : Unidade de disco atual da estação (DEFAULT).**
- 1 : Drive A: da estação remota.**
- 2 : Drive B: da estação remota.**
- 3 : Drive C: da estação remota.**
- 4 : Drive D: da estação remota ... e assim por diante.**

Caso a função DiskSpace seja executada através de um Job ( processo isolado no Servidor , sem interface Remota ) , ou seja passado um argumento de unidade de disco inexistente ou indisponível , a função DISKSPACE() retornará -1

**Sintaxe: DISKSPACE ( [ nDrive ] )**

**Parâmetros:**

<b>nDrive</b>	Número do drive, onde 0 é o espaço na unidade de disco corrente, e 1 é o drive A: do cliente, 2 é o drive B: do cliente, etc.
---------------	---

**Retorno:**

<b>Numérico</b>	Número de bytes disponíveis no disco informado como parâmetro.
-----------------	--

### **Exemplo:**

```
nBytesLocal := DISKSPACE( ) // Retorna o espaço disponível na unidade de disco local  
  
IF nBytesLocal < 1048576  
    MsgStop('Unidade de Disco local possui menos de 1 MB livre.')  
Else  
    MsgStop('Unidade de disco local possui '+str(nBytes_A,12)+' bytes livres.')  
Endif  
nBytes_A := DISKSPACE( 1 ) // Retorna o espaço disponível no drive A: local ( remote ).  
  
If nBytes_A == -1  
    MsgStop('Unidade A: não está disponível ou não há disco no Drive')  
ElseIf nBytes_A < 8192  
    MsgStop('Não há espaço disponível no disco. Substitua o disco na Unidade A:')  
Else  
    MsgStop('Unidade A: Verificada . '+str(nBytes_A,12)+' bytes livres.')  
Endif
```

## **EXISTDIR()**

Função utilizada para determinar se um path de diretório existe e é valido.

- Sintaxe: EXISTDIR (< cPath >)**

- Parâmetros:**

<b>cPath</b>	String contendo o diretório que será verificado, caso seja feita uma verificação a partir do server, devemos informar a partir do rootPath do Protheus, caso contrário devemos passar o path completo do diretório.
--------------	---

- Retorno:**

<b>Lógico</b>	Retorna se verdadeiro(.T.) caso o diretório solicitado exista, falso(.F.) caso contrário.
---------------	---

### **Exemplo 01: No server a partir do rootPath**

```
lRet := ExistDir('\teste')
```

### **Exemplo 02: No client, passando o FullPath**

```
lRet := ExistDir('c:\APO')
```

## **FCLOSE()**

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

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

- Sintaxe: FCLOSE ( < nHandle > )**

- Parâmetros:**

<b>nHandle</b>	Handle do arquivo obtido previamente através de FOPEN() ou FCREATE().
----------------	---

- Retorno:**

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

**Exemplo:**

```
#include "Fileio.ch"

nHandle := FCREATE("Testfile", FC_NORMAL)

If !FCLOSE(nHandle)
    conout( "Erro ao fechar arquivo, erro numero: ", FERROR() )
EndIf
```

**FCREATE()**

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

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

 **Atributos definidos no include FileIO.ch**

Constante	Valor	Descrição
FC_NORMAL	<b>0</b>	Criação normal do Arquivo (default/padrão).
FC_READONLY	<b>1</b>	Cria o arquivo protegido para gravação.
FC_HIDDEN	<b>2</b>	Cria o arquivo como oculto.
FC_SYSTEM	<b>4</b>	Cria o arquivo como sistema.

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

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

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

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

 **Retorno:**

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

## **FERASE()**

Função utilizada para apagar um arquivo no disco . O Arquivo pode estar no Servidor ou na estação local (Remote). O arquivo para ser apagado deve estar fechado, não sendo permitido a utilização de caracteres coringa (wildcards).

- Sintaxe:** FERASE ( < cArquivo > )

- Parâmetros:**

<b>cArquivo</b>	Nome do arquivo a ser apagado . Pode ser especificado um path absoluto ou relativo , para apagar arquivos na estação local ( Remote ) ou no Servidor, respectivamente.
-----------------	--

- Retorno:**

<b>Numérico</b>	A função retornará 0 caso o arquivo seja apagado com sucesso , e -1 caso não seja possível apagar o arquivo. Caso a função retorne -1, é possível obter maiores detalhes da ocorrência através da função FERROR().
-----------------	--

### **Exemplo:**

```
#include 'DIRECTRY.CH'

aEval(Directory("*.BAK"), { |aFile| FERASE(aFile[F_NAME]) })

// Este exemplo apaga um arquivo no cliente ( Remote ) , informando o status da operação
IF FERASE("C:\ListaTXT.tmp") == -1
    MsgStop('Falha na deleção do Arquivo ( FError'+str(ferror(),4)+ ')')
Else
    MsgStop('Arquivo deletado com sucesso.')
ENDIF
```

## **FILE()**

Função que verifica se existe um arquivo ou um padrão de arquivos, no diretório. Podem ser especificados caminhos absolutos ( arquivos na estação - Remote ) ou relativos ( a partir do RootPath do Protheus Server) , sendo os caracteres "\*" e "?" ( wildcards ) aceitos.

- Sintaxe:** FILE ( < cArquivo > )

- Parâmetros:**

<b>cArquivo</b>	Nome do arquivo , podendo ser especificado um path (caminho) . Caminhos locais (Remote) ou caminhos de servidor são aceitos , bem como wildcards (Caracteres "*" e "?").
-----------------	--

- Retorno:**

<b>Lógico</b>	O retorno será .T. caso o arquivo especificado exista. Caso o mesmo não exista no path especificado , a função retorna .F.
---------------	--

**Exemplo:**

```
//Verifica no diretório corrente do servidor se existe o arquivo teste.dbf  
FILE("teste.dbf")  
  
// Verifica no diretório Sigaadv do servidor se existe o arquivo teste.dbf  
FILE("\SIGAADV\TESTE.dbf")  
  
// Verifica no diretório Temp do cliente (Remote) se existe o arquivo teste.dbf  
FILE("C:\TEMP\TESTE.dbf")
```



*Importante*

Caso a função FILE() seja executada em Job ( programa sem interface remota ), sendo passado um caminho absoluto de arquivo ( exemplo c:\teste.txt ) , a função retornará .F. e FERROR() retornará -1 ).

**FILENOEXT()**

Função que retorna o nome de um arquivo contido em uma string, ignorando a extensão.

**Sintaxe: FileNoExt( cString )**

**Parâmetros**

<b>cString</b>	String contendo o nome do arquivo.
----------------	------------------------------------

**Exemplo:**

```
Local cString := '\SIGAADV\ARQZZZ.DBF'  
cString := FileNoExt( cString )  
// Retorno → "\SIGAADV\ARQZZZ"
```



*Anotações*

---

---

---

---

## **FOPEN()**

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

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

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



**Importante**

- FOPEN procura o arquivo no diretório corrente e nos diretórios configurados na variável de pesquisa do Sistema Operacional, a não ser que um path seja declarado explicitamente como parte do argumento <cArq>.
- Por serem executadas em um ambiente cliente-servidor, as funções de tratamento de arquivos podem trabalhar em arquivos localizados no cliente (estação) ou no servidor. O ADVPL identifica o local onde o arquivo será manipulado através da existência ou não da letra do drive no nome do arquivo passado em <cArq>. Ou seja, se o arquivo for especificado com a letra do drive, será aberto na estação. Caso contrário, será aberto no servidor com o diretório configurado como rootpath sendo o diretório raiz para localização do arquivo.

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

**Parâmetros:**

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

**Retorno:**

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

**Exemplo:**

```
#include 'fileio.ch'  
...  
nH := fopen('\sigaadv\error.log' , FO_READWRITE + FO_SHARED )  
If nH == -1  
    MsgStop('Erro de abertura : FERROR '+str(ferror(),4))  
Else  
    MsgStop('Arquivo aberto com sucesso.')  
...  
fclose(nH)  
Endif  
...
```

**Tabela A: Modos de acesso a arquivos binários**

Modo	Constate(fileio.ch)	Operação
<b>0</b>	FO_READ	Aberto para leitura (padrão assumido)
<b>1</b>	FO_WRITE	Aberto para gravação
<b>2</b>	FO_READWRITE	Aberto para leitura e gravação

**Tabela B: Modos de acesso de compartilhamento a arquivos binários**

Modo	Constate(fileio.ch)	Operação
<b>0</b>	FO_COMPAT	Modo de Compatibilidade (Default)
<b>16</b>	FO_EXCLUSIVE	Acesso total exclusivo
<b>32</b>	FO_DENYWRITE	Acesso bloqueando a gravação de outros processos ao arquivo.
<b>48</b>	FO_DENYREAD	Acesso bloqueando a leitura de outros processos ao arquivo.
<b>64</b>	FO_DENYNONE	Acesso compartilhado. Permite a leitura e gravação por outros.



*Anotações*

---

---

---

---

## **FREAD()**

---

Função que realiza a leitura dos dados a partir um arquivo aberto, através de FOPEN(), FCREATE() e/ou FOPENPORT(), e armazena os dados lidos por referência no buffer informado. FREAD() lerá até o número de bytes informado em nQtdBytes; caso aconteça algum erro ou o arquivo chegue ao final, FREAD() retornará um número menor que o especificado em nQtdBytes. FREAD() lê normalmente caracteres de controle (ASC 128, ASC 0, etc.) e lê a partir da posição atual do ponteiro atual do arquivo , que pode ser ajustado ou modificado pelas funções FSEEK() , FWRITE() ou FREADSTR().

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

- Sintaxe: FREAD ( < nHandle > , < cBuffer > , < nQtdBytes > )**

**Parâmetros:**

<b>nHandle</b>	É o manipulador (Handle) retornado pelas funções FOPEN(), FCREATE(), FOPENPORT(), que faz referência ao arquivo a ser lido.
<b>cBuffer</b>	É o nome de uma variável do tipo String , a ser utilizada como buffer de leitura , onde os dados lidos deverão ser armazenados. O tamanho desta variável deve ser maior ou igual ao tamanho informado em nQtdBytes. Esta variável deve ser sempre passada por referência. ( @ antes do nome da variável ), caso contrário os dados lidos não serão retornados.
<b>nQtdBytes</b>	Define a quantidade de Bytes que devem ser lidas do arquivo a partir posicionamento do ponteiro atual.

**Retorno:**

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

## **FREADSTR ()**

---

Função que realiza a leitura de caracteres de um arquivo binário. FREADSTR() lê de um arquivo aberto, através de FOPEN(), FCREATE(), FOPENPORT(). FREADSTR() lerá até o número de bytes informado em nQtdBytes ou até encontrar um CHR(0). Caso aconteça algum erro ou o arquivo chegue ao final, FREADSTR() retornará uma string menor do que nQdBytes e colocará o erro em FERROR(). FREADSTR() lê a partir da posição atual do ponteiro, que pode ser ajustado pelo FSEEK(), FWRITE( ) ou FREAD().

- Sintaxe: FREADSTR ( < nHandle > , < nQtdBytes > )**

**Parâmetros:**

<b>nHandle</b>	É o manipulador retornado pelas funções FOPEN(), FCREATE(), FOPENPORT().
<b>nQtdBytes</b>	Número máximo de bytes que devem ser lidos.

**Retorno:**

<b>Caracter</b>	Retorna uma string contendo os caracteres lidos.
-----------------	--

## **FRENAME()**

Através da função FRENAME() é possível renomear um arquivo para outro nome, tanto no servidor como na estação. Ao renomear um arquivo não esqueça que esta arquivo deverá estar fechado ( isto é , não pode estar em uso por nenhum outro processo ou estação). Caso o arquivo esteja aberto por outro processo , a operação de renomear o arquivo não é possível. A função fRename() não aceita wildcards (\* e/ou ? ).

Vale lembrar que não é possível renomear um arquivo especificando nos parâmetros simultaneamente um caminho de servidor e um de estação remota, bem como especificar dois arquivos remotos e executar a função fRename() através de um JOB. Caso isto ocorra, a função retornará -1 , e fError() retornará também -1.



*Importante*

Quando especificamos um path diferente nos arquivos de origem e destino , a função fRename() realiza a funcionalidade de MOVER o arquivo para o Path especificado.

- Sintaxe: FRENAME ( < cOldFile > , < cNewFile > )**

- Parâmetros:**

<b>cOldFile</b>	Nome do arquivo será renomeado, aceita caminhos do servidor e caminhos do cliente. Caso não seja especificado nenhuma unidade de disco e path, é considerado o path atual no servidor.
<b>cNewFile</b>	Novo nome do arquivo, aceita também caminho do servidor, e caminho do cliente.

- Retorno:**

<b>Numérico</b>	Se o status retornado for -1 , ocorreu algum erro na mudança de nome : Verifique se os dois caminhos estão no mesmo ambiente, verifique a existência do arquivo de origem, se ele não está em uso no momento por outro processo , e verifique se o nome do arquivo de destino já não existe no path de destino especificado.
-----------------	--



*Anotações*

---

---

---

---

---

## **FSEEK()**

---

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

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

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

- Sintaxe: FSEEK ( < nHandle > , [ nOffSet ] , [ nOrigem ] )**

- Parâmetros:**

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

- Retorno:**

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

## **FT\_FEOF()**

---

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

- Sintaxe: FT\_FEOF( )**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

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

## **FT\_FGOTO()**

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

- Sintaxe:** FT\_FGOTO ( < nPos > )

- Parâmetros:**

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

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **FT\_FGOTOP()**

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

- Sintaxe:** FT\_FGOTO ( < nPos > )

- Parâmetros:**

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

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **FT\_FLASTREC()**

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



Dica

Verifique maiores informações sobre formato do arquivo e tamanho máximo da linha de texto na função FT\_FREADLN().

- Sintaxe:** FT\_FLASTREC( )

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

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

**Exemplo:**

```
FT_FUse('teste.txt') // Abre o arquivo  
CONOUT("Linhas no arquivo ["+str(ft_flastrec(),6)+""]")  
FT_FGOTOP()  
While !FT_FEof()  
    conout("Ponteiro ["+str(FT_FRECNO(),6)+""] Linha ["+FT_FReadln()+""]")  
    FT_FSkip()  
Enddo  
FT_FUse() // Fecha o arquivo
```

**FT\_FREADLN()**

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



*Importante*

- A utilização desta função não altera a posição do ponteiro para leitura dos dados, o ponteiro do arquivo não é movido. A movimentação do ponteiro é realizada através da função FT\_FSKIP()
- O limite de 1022 bytes por linha inclui os caracteres delimitadores de final de linha. Deste modo, quando utilizados os separadores CRLF, isto nos deixa 1020 bytes de texto, e utilizando LF, 1021 bytes. A tentativa de leitura de arquivos com linhas de texto maiores do que os valores especificados acima resultará na leitura dos 1023 primeiros bytes da linha, e incorreta identificação das quebras de linha posteriores.
- As funções FT\_F\* foram projetadas para ler arquivos com conteúdo texto apenas. A utilização das mesmas em arquivos binários pode gerar comportamentos inesperados na movimentação do ponteiro de leitura do arquivo, e incorretas identificações nos separadores de final de linha.



*Dica*

- **Release:** Quando utilizado um Protheus Server, com build superior a 7.00.050713P, a função FT\_FREADLN() também é capaz de ler arquivos texto / ASCII, que utilizam também o caractere LF (*chr(10)*) como separador de linha.

 **Sintaxe: FT\_FREADLN( )** **Parâmetros:**

<b>Nenhum</b>	.
---------------	---

 **Retorno:**

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

## **FT\_FRECNO()**

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

A função FT\_FRecno retorna a posição corrente do ponteiro do arquivo texto aberto pela FT\_FUse.

- Sintaxe:** **FT\_FRECNO ( )**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Caracter</b>	Retorna a posição corrente do ponteiro do arquivo texto.
-----------------	--

## **FT\_FSKIP()**

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

- Sintaxe:** **FT\_FSKIP ( [ nLinhas ] )**

- Parâmetros:**

<b>nLinhas</b>	nLinhas corresponde ao número de linhas do arquivo TXT ref. movimentação do ponteiro de leitura do arquivo.
----------------	---

- Retorno**

<b>Nenhum</b>	.
---------------	---

## **FT\_FUSE()**

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



Dica

Verifique maiores informações sobre formato do arquivo e tamanho máximo da linha de texto na função FT\_FREADLN().

- Sintaxe:** **FT\_FUSE ( [ cTXTFile ] )**

- Parâmetros:**

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

- Retorno:**

<b>Numérico</b>	A função retorna o Handle de controle do arquivo. Em caso de falha de
-----------------	---

## FWRITE()

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

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

- Sintaxe:** **FWRITE ( < nHandle > , < cBuffer > , [ nQtdBytes ] )**

**Parâmetros:**

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

**Retorno:**

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

**Exemplo:**

```
#INCLUDE "FILEIO.CH"
#define F_BLOCK 1024 // Define o bloco de Bytes a serem lidos / gravados por vez

User Function TestCopy()
Local cBuffer := SPACE(F_BLOCK)
Local nHOrigem , nHDestino
Local nBytesLidos , nBytesFalta , nTamArquivo
Local nBytesLer , nBytesSalvo
Local lCopiaOk := .T.

// Abre o arquivo de Origem
nHOrigem := FOPEN("ORIGEM.TXT", FO_READ)
```

**Exemplo (continuação):**

```
// Testa a abertura do Arquivo
If nHOrigem == -1
    MsgStop('Erro ao abrir origem. Ferror = '+str(ferror(),4),'Erro')
    Return .F.
Endif

// Determina o tamanho do arquivo de origem
nTamArquivo := Fseek(nHOrigem,0,2)

// Move o ponteiro do arquivo de origem para o inicio do arquivo
Fseek(nHOrigem,0)

// Cria o arquivo de destino
nHDestino := FCREATE("DESTINO.TXT", FC_NORMAL)

// Testa a criação do arquivo de destino
If nHDestino == -1
    MsgStop('Erro ao criar destino. Ferror = '+str(ferror(),4),'Erro')
    FCLOSE(nHOrigem) // Fecha o arquivo de Origem
    Return .F.
Endif

// Define que a quantidade que falta copiar é o próprio tamanho do Arquivo
nBytesFalta := nTamArquivo

// Enquanto houver dados a serem copiados
While nBytesFalta > 0

    // Determina quantidade de dados a serem lidos
    nBytesLer := Min(nBytesFalta , F_BLOCK )

    // lê os dados do Arquivo
    nBytesLidos := FREAD(nHOrigem, @cBuffer, nBytesLer )

    // Determina se não houve falha na leitura
    If nBytesLidos < nBytesLer
        MsgStop( "Erro de Leitura da Origem. "+;
                  Str(nBytesLer,8,2)+" bytes a LER."+;
                  Str(nBytesLidos,8,2)+" bytes Lidos."+;
                  "Ferror = "+str(ferror(),4),'Erro')
        lCopiaOk := .F.
        Exit
    Endif

    // Salva os dados lidos no arquivo de destino
    nBytesSalvo := FWRITE(nHDestino, cBuffer,nBytesLer)

    // Determina se não houve falha na gravação
    If nBytesSalvo < nBytesLer
        MsgStop("Erro de gravação do Destino. "+;
                  Str(nBytesLer,8,2)+" bytes a SALVAR."+;
                  Str(nBytesSalvo,8,2)+" bytes gravados."+;
                  "Ferror = "+str(ferror(),4),'Erro')
        lCopiaOk := .F.
        EXIT
    Endif
```

**Exemplo (continuação) :**

```
// Elimina do Total do Arquivo a quantidade de bytes copiados  
nBytesFalta -= nBytesLer  
  
Enddo  
  
// Fecha os arquivos de origem e destino  
FCLOSE(nHOrigem)  
FCLOSE(nHDestino)  
  
If lCopiaOk  
    MsgStop('Cópia de Arquivos finalizada com sucesso. '+;  
            str(nTamArquivo,12,0)+' bytes copiados.', 'Final')  
Else  
    MsgStop( 'Falha na Cópia. Arquivo de Destino incompleto. '+;  
             'Do total de '+str(nTamArquivo,12,0)+' bytes, faltaram  
'+str(nBytesFalta,12,0)+' bytes.', 'Final')  
Endif  
  
Return
```

**MSCOPYFILE()**

Função que executa a cópia binária de um arquivo para o destino especificado.

**Sintaxe: MSCOPYFILE( cArqOrig, cArqDest )**

**Parâmetros:**

<b>cArqOrig</b>	Nome do arquivo origem e a extensão.
<b>cArqDest</b>	Nome do arquivo destino e a extensão.

**Retorno:**

<b>Lógico</b>	Se a copia for realizada com sucesso a função retornará verdadeiro (.T.).
---------------	---

**Exemplo:**

```
Local cArqOrig := 'ARQ00001.DBF'  
Local cArqDest := 'ARQ00002.XXX'  
  
If MsCopyFile( cArqOrig, cArqDest )  
    APMsgInfo('Copia realizada com sucesso!')  
EndIf
```

## **MSCOPYTO()**

Função que realiza a cópia dos registros de uma base de dados para outra, criando o arquivo destino de acordo com a estrutura da base de dados origem.

- Sintaxe:** **MSCOPYTO( [cArqOrig], cArqDest )**

- Parâmetros:**

<b>cArqOrig</b>	Nome do arquivo origem e a extensão se o ambiente for Top o parâmetro passará a ser obrigatório.
<b>cArqDest</b>	Nome do arquivo destino e a extensão.

- Retorno:**

<b>Lógico</b>	Se a copia for realizada com sucesso a função retornará verdadeiro (.T.).
---------------	---

### **Exemplo:**

```
Local cArqDest := 'SX2ZZZ.DBF'
DbSelectArea('SX2')
If MsCopyTo( , cArqDest )
    APMsgInfo('Copia realizada com sucesso!')
Else
    APMsgInfo('Problemas ao copiar o arquivo SX2!')
EndIf
```

## **MSCREATE()**

Função que cria um arquivo (tabela) de acordo com a estrutura informada no parâmetro aStruct. Se o parâmetro cDriver não for informado o RDD corrente será assumido como padrão. Para criação de tabelas no TopConnect é necessário estar conectado ao banco e o environment do protheus ser TOP.



*Importante*

aStruct: array contendo a estrutura da tabela aonde:  
 1º - caracter, nome do campo;  
 2º - caracter, tipo do campo;  
 3º - numérico, tamanho do campo;  
 4º - numérico, decimais.

- Sintaxe:** **MsCreate( cArquivo, aStru ,[cDriver] )**

- Parâmetros:**

<b>cArquivo</b>	Nome do arquivo.
<b>aStruct</b>	Estrutura do arquivo.
<b>cDriver</b>	RDD do arquivo.

- Retorno:**

<b>Lógico</b>	Indica se a operação foi executada com sucesso.
---------------	---

**Exemplo:**

```
Local cTarget := '\sigaadv\'  
Local aStrut  
aStrut := { { 'Campo', 'C', 40, 0 } }  
If MsCreate( cTarget+'ARQ1001', aStrut )  
    APMsgInfo('Criado com sucesso!')  
Else  
    APMsgInfo('Problemas ao criar o arquivo!')  
EndIf
```

**MSERASE()**

Função utilizada para deletar fisicamente o arquivo especificado.

- Sintaxe: MsErase( cArquivo, [cIndice], [cDriver] )**

- Parâmetros:**

<b>cArquivo</b>	Nome do arquivo e a extensão.
<b>cIndice</b>	Nome do arquivo de índice e a extensão.
<b>cDriver</b>	RDD do arquivo, se não for informado assumirá o RDD corrente.

- Retorno:**

<b>Lógico</b>	Indica se a operação foi executada com sucesso.
---------------	---

**Exemplo:**

```
Local cArquivo := 'SX2ZZZ.DBF'  
Local cIndice := 'SX2ZZZ'+ OrdBagExt()  
If MsErase( cArquivo, cIndice )  
    APMsgInfo( 'Arquivo deletado com sucesso!' )  
Else  
    APMsgInfo( 'Problemas ao deletar arquivo!' )  
EndIf
```



Anotações

---

---

---

---

## **MSRENAME()**

Função que verifica a existência do arquivo especificado.

- Sintaxe:** MsFile( **cArquivo**, [cIndice], [cDriver] )

- Parâmetros:**

<b>cArquivo</b>	Nome do arquivo e a extensão.
<b>cIndice</b>	Nome do arquivo de índice e a extensão.
<b>cDriver</b>	RDD do arquivo, se não for informado assumirá o RDD corrente.

- Retorno:**

<b>Lógico</b>	Indica se o arquivo especificado existe.
---------------	--

### **Exemplo:**

```
Local cArquivo := 'SX2ZZZ.DBF'  
Local cIndice := 'SX2ZZZ'+ OrdBagExt()  
If !MsFile ( cArquivo, cIndice )  
    APMsgInfo( 'Arquivo não encontrado!' )  
EndIf
```

## **RETFILENAME()**

Função que retorna o nome de um arquivo contido em uma string, ignorando o caminho e a extensão.

- Sintaxe:** RetFileName( **cArquivo** )

- Parâmetros:**

<b>cArquivo</b>	String contendo o nome do arquivo
-----------------	-----------------------------------

- Retorno:**

<b>Caracter</b>	Nome do arquivo contido na string cArquivo sem o caminho e a extensão.
-----------------	--

### **Exemplo:**

```
Local cArquivo := '\SIGAADV\ARQZZZ.DBF'  
cArquivo := RetFileName( cArquivo )  
// retorno 'ARQZZZ'
```

## Manipulação de arquivos e índices temporários

### CRIATRAB()

Função que cria um arquivo de trabalho com uma estrutura especificada, sendo que:

- Caso o parâmetro IDbf seja definido como .T., a função criará um arquivo DBF com este nome e a estrutura definida em aArray.
- Caso o parâmetro IDbf seja definido como .F., a função não criará arquivo de nenhum tipo, apenas fornecerá um nome válido.

**Sintaxe: CriaTrab(aArray, IDbf)**

**Parâmetros:**

<b>aArray</b>	Array multidimensional contendo a estrutura de campos da tabela que será criada no formato: {Nome, Tipo, Tamanho, Decimal}
<b>IDbf</b>	Determina se o arquivo de trabalho deve ser criado ( .T.) ou não (.F. )

**Retorno:**

<b>Caracter</b>	Nome do Arquivo gerado pela função.
-----------------	-------------------------------------

**Exemplo:**

```
// Com IDbf = .F.  
cArq := CriaTrab(NIL, .F.)  
cIndice := "C9_AGREG+"+IndexKey()  
Index on &cIndice To &cArq  
  
// Com IDbf = .T.  
aStru := {}  
AADD(aStru, { "MARK" , "C", 1, 0})  
AADD(aStru, { "AGLUT" , "C", 10, 0})  
AADD(aStru, { "NUMOP" , "C", 10, 0})  
AADD(aStru, { "PRODUTO" , "C", 15, 0})  
AADD(aStru, { "QUANT" , "N", 16, 4})  
AADD(aStru, { "ENTREGA" , "D", 8, 0})  
AADD(aStru, { "ENTRAJU" , "D", 8, 0})  
AADD(aStru, { "ORDEM" , "N", 4, 0})  
AADD(aStru, { "GERADO" , "C", 1, 0})  
cArqTrab := CriaTrab(aStru, .T.)  
USE &cArqTrab ALIAS TRB NEW
```



*Importante*

Na criação de índices de trabalho temporários é utilizada a sintaxe:

**CriaTrab(Nil, .F.)**

## **Manipulação de bases de dados**

### **ALIAS()**

Função de banco de dados utilizada para determinar o alias da área de trabalho especificada. Alias é o nome atribuído a uma área de trabalho quando um arquivo de banco de dados está em uso. O nome real atribuído é o nome do arquivo de banco de dados, ou um nome que foi explicitamente atribuído através da cláusula ALIAS do comando USE.

A função ALIAS() é o inverso da função SELECT() pois retorna o alias através do número da área de trabalho, enquanto SELECT() retorna o número da área de trabalho através do alias.

- Sintaxe:** ALIAS ( [ nAreaTrabalho ] )

- Parâmetros:**

<b>nAreaTrabalho</b>	<nAreaTrabalho> é o número da área de trabalho a ser verificada.
----------------------	--

- Retorno:**

<b>Caracter</b>	Retorna o alias da área de trabalho especificada na forma de uma cadeia de caracteres, em letra maiúscula. Caso <nAreaTrabalho> não seja especificada, é retornado o alias da área de trabalho corrente. Se não houver nenhum arquivo de banco de dados em USo na área de trabalho especificada, ALIAS() retorna uma cadeia de caracteres nula ("").
-----------------	--

### **Exemplo:**

```
cAlias := alias()
IF empty(cAlias)
    alert('Não há Área em uso')
Else
    alert(Area em uso atual : '+cAlias)
Endif
```

### **BOF() / EOF()**

As funções BOF() e EOF() são utilizadas para determinar se o ponteiro de leitura do arquivo encontra-se no começo ou no final do mesmo conforme abaixo:

- BOF() é uma função de tratamento de banco de dados utilizada para testar uma condição de limite de inicial do arquivo quando o ponteiro de registros está se movendo para trás em um arquivo de banco de dados.
- EOF() é uma função de tratamento de banco de dados utilizada para testar uma condição de limite de final de arquivo quando o ponteiro de registros está se movendo para frente em um arquivo de banco de dados.

Normalmente é utilizada a condição EOF() como parte do argumento <lCondicao> de uma construção DO WHILE que processa registros sequencialmente em um arquivo de banco de dados. Neste caso <lCondicao> incluiria um teste para .NOT. EOF(), forçando o laço DO WHILE a terminar quando EOF() retornar verdadeiro (.T.)

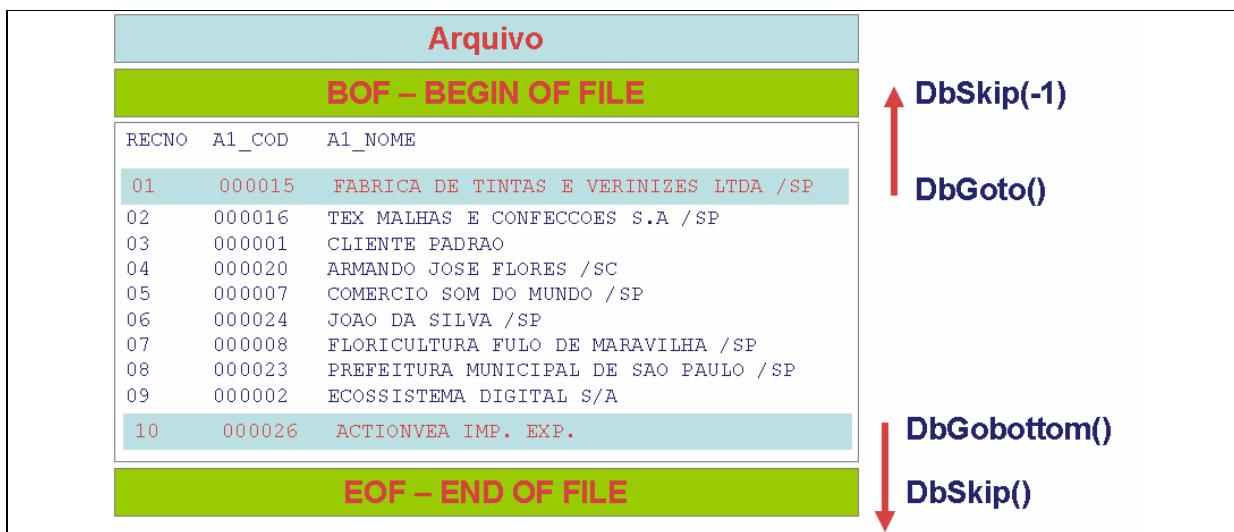
**Sintaxe: BOF() / EOF()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Lógico</b>	Retorna verdadeiro (.T.) quando , feita uma tentativa de mover o ponteiro de registros para além do primeiro registro lógico em um arquivo de banco de dados, do contrário, ela retorna falso (.F.).
<b>Lógico</b>	Retorna verdadeiro (.T.) quando , feita uma tentativa de mover o ponteiro de registros para além do último registro lógico em um arquivo de banco de dados, do contrário, ela retorna falso (.F.). Caso não haja nenhum arquivo de banco de dados aberto na área de trabalho corrente, EOF() retorna falso (.F.). Se o arquivo de banco de dados corrente não possui registros, EOF() retorna verdadeiro (.T.).



## COPY()

O comando COPY TO permite a cópia de todos ou parte dos registros da tabela atualmente selecionada como área de trabalho atual, para um novo arquivo. Os registros considerados para a cópia podem ser limitados pela cláusula <escopo>, através de expressões FOR WHILE, e/ou através de um filtro.

Se o filtro para registros deletados ( SET DELETED ) estiver desligado (OFF), registros deletados ( marcados para deleção ) são copiados para o arquivo de destino, mantendo este status. Caso contrário, nenhum registro deletado é copiado. Da mesma maneira, caso exista uma condição de filtro na tabela atual ( SET FILTER ), apenas os registros que satisfaçam a condição de filtro serão copiados.

Os registros são lidos na tabela atual, respeitando a ordem de índice setada. Caso não hajam índices abertos, ou a ordem de navegação nos índices (SET ORDER ) seja 0 (zero), os registros são lidos em orden natural ( ordem de RECNO ).

A tabela de destino dos dados copiados é criada, e aberta em modo exclusivo, antes da operação de cópia efetiva ser iniciada.

**Tabela A : Especificação do formato SDF ( System Data Format )**

<b>Elemento do Arquivo</b>	<b>Formato</b>
<b>Campos 'C' Caractere</b>	Tamanho fixo, ajustado com espaços em branco
<b>Campos 'D' Data</b>	Formato aaaammdd ( ano, mês, dia )
<b>Campos 'L' lógicos</b>	T ou F
<b>Campos 'M' Memo</b>	(campo ignorado)
<b>Campos 'N' Numéricos</b>	Ajustados à direita, com espaços em branco.
<b>Delimitador de Campos</b>	Nenhum
<b>Separador de Registros</b>	CRLF ( ASCII 13 + ASCII 10 )
<b>Marca de final de arquivo (EOF)</b>	Nenhum

**Tabela B : Especificação do formato delimitado ( DELIMITED / DELIMITED WITH <cDelimiter> )**

<b>Elemento do Arquivo</b>	<b>Formato</b>
<b>Campos 'C' Caractere</b>	Delimitados, ignorando espaços à direita
<b>Campos 'D' Data</b>	Formato aaaammdd ( ano, mês, dia )
<b>Campos 'L' lógicos</b>	T ou F
<b>Campos 'M' Memo</b>	(campo ignorado)
<b>Campos 'N' Numéricos</b>	sem espaços em branco.
<b>Delimitador de Campos</b>	Vírgula
<b>Separador de Registros</b>	CRLF ( ASCII 13 + ASCII 10 )
<b>Marca de final de arquivo (EOF)</b>	Nenhum



*Importante*

A Linguagem Advpl, antes do Protheus, suportava a geração de uma tabela delimitada diferenciada, obtida através do comando *COPY TO (...) DELIMITED WITH BLANK*. No Protheus este formato não é suportado. Caso utilize-se este comando com a sintaxe acima, o arquivo ASCII gerado será delimitado, utilizando-se a sequência de caracteres 'BLANK' como delimitadora de campos Caractere.

**Sintaxe:**

```
COPY [ FIELDS <campo,...> ] TO cFile [cEscopo] [ WHILE <ICondicao> ]
[ FOR <ICondicao> ] [ SDF | DELIMITED [WITH <cDelimiter>] ]
[ VIA <cDriver> ]
```

**Parâmetros:**

<b>FIELDS &lt;campo,...&gt;</b>	FIELDS <campo,...> especifica um ou mais campos, separados por vírgula, a serem copiados para a tabela de destino. Caso não especificado este parâmetro, serão copiados todos os campos da tabela de origem.
<b>TO cFile</b>	TO <cFile> especifica o nome do arquivo de destino. O nome do arquivo de destino pode ser especificado de forma literal direta, ou como uma expressão Advpl, entre parênteses.  Caso sejam especificadas as cláusulas SDF ou DELIMITED, é gerado um arquivo ASCII, com extensão .txt por default.
<b>cEscopo</b>	<cEscopo> define a porção de dados da tabela atual a ser copiada. Por default, são copiados todos os registros (ALL). Os escopos possíveis de uso são:  ALL - Copia todos os registros. REST - Copia, a partir do registro atualmente posicionado, até o final da tabela. NEXT <n> - Copia apenas <n> registros, iniciando a partir do registro atualmente posicionado.  OBSERVAÇÃO : Vale a pena lembrar que o escopo é sensível também às demais condições de filtro ( WHILE / FOR ).
<b>WHILE &lt;lCondicao&gt;</b>	WHILE <lCondicao> permite especificar uma condição para realização da cópia, a partir do registro atual, executada antes de inserir cada registro na tabela de destino, sendo realizada a operação de cópia enquanto esta condição for verdadeira.
<b>FOR &lt;lCondicao&gt;</b>	FOR <lCondicao> especifica uma condição para cópia de registros, executada antes de inserir um registro na tabela de destino, sendo a operação realizada apenas se lCondicao ser verdadeira ( .T. )
<b>[SDF DELIMITED]</b>	[ SDF   DELIMITED [WITH <xcDelimiter>] ]  SDF especifica que o tipo de arquivo de destino gerado é um arquivo no formato "System Data Format" ASCII, onde registros e campos possuem tamanho fixo no arquivo de destino.  DELIMITED especifica que o arquivo ASCII de destino será no formato delimitado, onde os campos do tipo Caractere são delimitados entre aspas duplas ( delimitador Default ). Registros e campos têm tamanho variável no arquivo ASCII.  DELIMITED WITH <xcDelimiter> permite especificar um novo caractere, ou sequência de caracteres, a ser utilizada como delimitador, ao invés do default ( aspas duplas ). O caractere delimitador pode ser escrito de forma literal, ou como uma expressão entre parênteses.  Nas Tabelas complementares A e B, na documentação do comando, são detalhadas as especificações dos formatos SDF e DELIMITED.
<b>VIA &lt;cDriver&gt;</b>	VIA <xcDriver> permite especificar o driver utilizado para criar a tabela de destino dos dados a serem copiados.  O Driver deve ser especificado como uma expressão caractere. Caso especificado como um valor literal direto, o mesmo deve estar entre aspas.

**Retorno:**

<b>Nenhum</b>	.
---------------	---

## **COPY STRUCTURE()**

O comando COPY STRUCTURE cria uma nova tabela vazia, com a estrutura da tabela ativa na área de trabalho atual. Se a tabela a ser criada já exista, a mesma é sobreescrita. A tabela de destino criada utiliza o mesmo RDD da tabela de origem ( tabela ativa na área de trabalho atual ).



*Importante*

A Linguagem Advpl, antes do Protheus, suportava a parametrização de uma lista de campos da tabela atual, para compor a estrutura da tabela de destino, através da cláusula *FIELDS <campo,...>*. Esta opção não é suportada no Protheus. Caso seja utilizada, o programa será abortado com a ocorrência de erro fatal : **'DBCopyStruct - Parameter <Fields> not supported in Protheus'**

- Sintaxe:**

<b>COPY STRUCTURE TO &lt;xcDataBase&gt;</b>
---

- Parâmetros:**

<b>TO &lt;xcDataBase&gt;</b>	Deve ser especificado em xcDatabase o nome da tabela a ser criada.
------------------------------	--

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **DBAPPEND()**

A função DBAPPEND() acrescenta mais um registro em branco no final da tabela corrente. Se não houver erro da RDD, o registro é acrescentado e bloqueado.

- Sintaxe:** DBAPPEND ( [ ILiberaBloqueios ] )

- Parâmetros:**

<b>ILiberaBloqueios</b>	Se o valor for .T., libera todos os registros bloqueados anteriormente (locks). Se for .F., todos os bloqueios anteriores são mantidos. Valor default: .T.
-------------------------	--

- Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
USE Clientes NEW
FOR i:=1 to 5
  DBAPPEND(.F.)
  NOME := "XXX"
  END : ="YYY"
NEXT
// Os 5 registros incluídos permanecem bloqueados
DBAPPEND()
// Todos os bloqueios anteriores são liberados
```

**DBCLEARALLFILTER()**

A função DBCLEARALLFILTER() salva as atualizações realizadas e pendentes de todas as tabelas e depois limpa as condições de filtro de todas as tabelas.

- Sintaxe: DBCLEARALLFILTER()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
USE Clientes NEW
DBSETFILTER( {|| Idade < 40}, 'Idade < 40') // Seta a expressão de filtro
...
DBCLEARALLFILTER()
// Limpa a expressão de filtro de todas as ordens
```



*Anotações*

---

---

---

---

## **DBCLEARFILTER()**

---

A função DBCLEARFILTER() salva as atualizações realizadas e pendentes na tabela corrente e depois limpa todas as condições de filtro da ordem ativa no momento. Seu funcionamento é oposto ao comando SET FILTER.

- Sintaxe: DBCLEARFILTER()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
USE Clientes NEW
DBSETFILTER( {|| Idade < 40}, "Idade < 40" ) // Seta a expressão de filtro
...
DBCLEARFILTER()
// Limpa a expressão de filtro
```

## **DBCLEARINDEX()**

---

A função DBCLEARINDEX() salva as atualizações pendentes na tabela corrente e fecha todos os arquivos de índice da área de trabalho. Por consequência, limpa todas as ordens da lista. Seu funcionamento é oposto ao comando SET INDEX.

- Sintaxe: DBCLEARINDEX()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
USE Clientes NEW
DBSETINDEX("Nome") // Abre o arquivo de índice "Nome"
...
DBCLEARINDEX()
// Fecha todos os arquivos de índices
```

## **DBCLOSEALL()**

---

A função DBCLOSEALL() salva as atualizações pendentes, libera todos os registros bloqueados e fecha todas as tabelas abertas (áreas de trabalho) como se chamassem DBCLOSEAREA para cada área de trabalho.

- Sintaxe: DBCLOSEALL()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
// Este exemplo demonstra como se pode utilizar o DBCLOSEALL para fechar a área de trabalho atual.  
USE Clientes NEW  
DBSETINDEX("Nome") // Abre o arquivo de índice "Nome"  
USE Fornecedores NEW  
DBSETINDEX("Idade") // Abre o arquivo de índice "Idade"  
...  
DBCLOSEALL() //Fecha todas as áreas de trabalho, todos os indices e ordens
```

---

## **DBCLOSEAREA()**

---

A função DBCLOSEAREA() permite que um alias presente na conexão seja fechado, o que viabiliza seu reuso em outro operação. Este comando tem efeito apenas no alias ativo na conexão, sendo necessária sua utilização em conjunto com o comando DbSelectArea().

- Sintaxe: DBCLOSEAREA()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
DbUserArea(.T., "DBFCDX", "\SA1010.DBF", "SA1DBF", .T., .F.)  
DbSelectArea("SA1DBF")  
MsgInfo("A tabela SA1010.DBF possui:" + STRZERO(RecCount(),6) + " registros.")  
DbCloseArea()
```

## **DBCOMMIT()**

A função DBCOMMIT() salva em disco todas as atualizações pendentes na área de trabalho corrente.

- Sintaxe: DBCOMMIT()**

- Parâmetros:**

**Nenhum**

.

- Retorno:**

**Nenhum**

.

### **Exemplo:**

```
USE Clientes NEW
DBGOTO(100)
Nome := "Jose"
USE Fornecedores NEW
DBGOTO(168)
Nome := "Joao"
DBCOMMIT() // Salva em disco apenas as alterações realizadas na tabela Fornecedores
```

## **DBCOMMITALL()**

A função DBCOMMITALL() salva em disco todas as atualizações pendentes em todas as áreas de trabalho.

- Sintaxe: DBCOMMITALL()**

- Parâmetros:**

**Nenhum**

.

- Retorno:**

**Nenhum**

.

### **Exemplo:**

```
USE Clientes NEW
DBGOTO(100)
Nome := "Jose"
USE Fornecedores NEW
DBGOTO(168)
Nome := "Joao"
DBCOMMITALL()
// Salva em disco as alterações realizadas nas tabelas Clientes e Fornecedores
```

## **DBCREATE()**

A função DBCREATE() é utilizada para criar um novo arquivo de tabela cujo nome está especificado através do primeiro parâmetro (cNome) e estrutura através do segundo (aEstrutura). A estrutura é especificada através de um array com todos os campos, onde cada campo é expresso através de um array contendo {Nome, Tipo, Tamanho, Decimais}.

- Sintaxe:** **DBCREATE** ( < cNOME > , < aESTRUTURA > , [ cDRIVER ] )

**Parâmetros:**

<b>cNOME</b>	Nome do arquivo a ser criado. Se contém pasta, ela se localiza abaixo do "RootPath". Se não, é criado por padrão no caminho formado por "RootPath"+"StartPath"
<b>aESTRUTURA</b>	Lista com as informações dos campos para ser criada a tabela.
<b>cDRIVER</b>	Nome da RDD a ser utilizado para a criação da tabela. Se for omitido será criada com a corrente.

**Retorno:**

**Nenhum**

.

### **Exemplo:**

```
// Este exemplo mostra como se pode criar novo arquivo através da função DBCREATE:  
LOCAL aEstrutura :={{Cod,N,3,0},  
                    {Nome,C,10,0},  
                    {Idade,N,3,0},  
                    {Nasc,D,8,0},  
                    {Pagto,N,7,2}}  
// Cria a tabela com o RDD corrente  
DBCREATE('\teste\cliente.dbf', aEstrutura)  
USE '\teste\cliente.dbf' VIA 'DBFCDX' NEW
```

### **Erros mais comuns:**



*Importante*

1. *DBCreate - Data base files can only be created on the server:* O nome do arquivo a ser criado não pode conter 'driver', pois, por convenção, ele seria criado na máquina onde o Remote está rodando.
2. *DBCreate - Invalid empty filename:* Nome do arquivo não foi especificado
3. *DBCreate - Field's name cannot be 'DATA':* Algumas RDD's não suportam este nome de campo. É uma palavra reservada.
4. *DBCreate - The length of Field's name must be at most 10:* Nome do campo não pode ter mais que 10 caracteres.
5. *DBCreate - Field's name must be defined:* Nome do campo não foi definido.
6. *DBCreate - Field's type is not defined:* Tipo do campo não foi definido.
7. *DBCreate - invalid Field's type:* Tipo do campo é diferente de 'C', 'N', 'D', 'M' ou 'L'.
8. *DBCreate - Invalid numeric field format:* Considerando 'len' o tamanho

total do campo numérico e 'dec' o número de decimais, ocorre este erro se:

- (*len* = 1) .and. (*dec* <> 0): Se o tamanho total é 1, o campo não pode ter decimais
- (*len*>1) .and. (*len*< *dec* + 2): Se o tamanho total é maior que 1, ele deve ser maior que o número de decimais mais 2, para comportar o separador de decimais e ter pelo menos um algarismo na parte inteira.

**Exemplo:** O número 12.45 poderia ser o valor de um campo com len=5 e dec=2 (no mínimo).



### Erros mais comuns:

Podem ocorrer também erros decorrentes de permissão e direitos na pasta onde se está tentando criar o arquivo ou por algum problema no banco de dados. Verifique as mensagens do servidor Protheus e do banco de dados.

## **DBCREATEINDEX()**

A função DBCREATEINDEX() é utilizada para criar um novo arquivo de índice com o nome especificado através do primeiro parâmetro, sendo que se o mesmo existir é deletado e criado o novo. Para tanto são executados os passos a seguir:

- ❑ Salva fisicamente as alterações ocorridas na tabela corrente;
- ❑ Fecha todos os arquivos de índice abertos;
- ❑ Cria o novo índice;
- ❑ Seta o novo índice como a ordem corrente;
- ❑ Posiciona a tabela corrente no primeiro registro do índice.

Com exceção do RDD CTREE, a tabela corrente não precisa estar aberta em modo exclusivo para a criação de índice, pois na criação de índices no Ctree é alterada a estrutura da tabela, precisando para isto a tabela estar aberta em modo exclusivo.

**Sintaxe:**   **DBCREATEINDEX(<cNOME>,   <cEXPCHAVE>,   [bEXPCHAVE],   [IUNICO])**

**Parâmetros:**

<b>cNOME</b>	Nome do arquivo de índice a ser criado.
<b>cEXPCHAVE</b>	Expressão das chaves do índice a ser criado na forma de string.
<b>bEXPCHAVE</b>	Expressão das chaves do índice a ser criado na forma executável.
<b>IUNICO</b>	Cria índice como único (o padrão é .F.).

**Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
// Este exemplo mostra como se pode criar novo arquivo de índice criando a ordem sobre os  
// campos Nome e End e não aceitará duplicação:
```

```
USE Cliente VIA "DBFCDX" NEW  
DBCREATEINDEX("\teste\ind2.cdx","Nome+End",{ || Nome+End },.T.)
```

### **DBDELETE()**

A função DBDELETE() marca o registro corrente como “apagado” logicamente(), sendo necessária sua utilização em conjunto com as funções RecLock() e MsUnLock().

Para filtrar os registro marcados do alias corrente pode-se utilizar o comando SET DELETED e para apagá-los fisicamente pode-se utilizar a função \_\_DBPACK().

- Sintaxe:** DBDELETE ( )

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
DbSelectArea("SA1")  
DbSetOrder(1) // A1_FILIAL + A1_COD + A1_LOJA  
DbSeek("01" + "900001" + "01") // Busca exata  
  
IF Found()  
    RecLock("SA1",.F.) // Define que será realizada uma alteração no registro posicionado  
    DbDelete() // Efetua a exclusão lógica do registro posicionado.  
    MsUnLock() // Confirma e finaliza a operação  
ENDIF
```



*Anotações*

---

---

---

---

## **DBF()**

A função DBF() verifica qual é o Alias da área de trabalho corrente. O Alias é definido quando a tabela é aberta através do parâmetro correspondente (DBUSEAREA()). Esta função é o inverso da função SELECT(), pois nesta é retornado o número da área de trabalho do Alias correspondente.

- Sintaxe:** **DBF()**
- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Caracter</b>	Retorna o Alias corrente. Caso não exista Alias corrente retorna "" (String vazia).
-----------------	---

### **Exemplo:**

```
dbUseArea( .T., "dbfcdxads", "\dadosadv609\sa1990.dbf", "SSS", .T., .F. )
MessageBox("O Alias corrente é: "+DBF(),"Alias", 0) //Resultado: "O Alias corrente é: SSS"
```

## **DBFIELDINFO()**

A função DBFIELDINFO() é utilizada para obter informações sobre determinado campo da tabela corrente. O tipo de informação (primeiro argumento) é escolhido de acordo com as constantes abaixo:

- Tabela A : Constantes utilizadas na parametrização da função**

<b>Constante</b>	<b>Descrição</b>	<b>Retorno</b>
<b>DBS_NAME</b>	Nome do campo.	Caracter
<b>DBS_DEC</b>	Número de casas decimais.	Numérico
<b>DBS_LEN</b>	Tamanho.	Numérico
<b>DBS_TYPE</b>	Tipo.	Caracter

A posição do campo não leva em consideração os campos internos do Protheus (Recno e Deleted).

- Sintaxe:** DBFIELDINFO ( < nINFOTIPO > , < nCAMPO > )

- Parâmetros:**

<b>nINFOTIPO</b>	Tipo de informação a ser verificada (DBS_NAME, DBS_DEC, DBS_LEN e DBS_TYPE).
<b>nCAMPO</b>	Posição do campo a ser verificado.

**Retorno:**

<b>Indefinido</b>	Retorna NIL se não há tabela corrente ou a posição do campo especificado está inválida. Informação do campo Informação requisitada pelo usuário (pode ser de tipo numérico se for tamanho ou casas decimais, tipo caracter se for nome ou tipo).
-------------------	---

**Exemplo:**

USE Clientes NEW

```
DBFIELDINFO(DBS_NAME,1) // Retorno: Nome  
DBFIELDINFO(DBS_TYPE,1) // Retorno: C  
DBFIELDINFO(DBS_LEN,1) // Retorno: 10  
DBFIELDINFO(DBS_DEC,1) // Retorno: 0
```

**DBFILTER()**

A função DBFILTER() é utilizada para verificar a expressão de filtro ativo na área de trabalho corrente.

**Sintaxe:** DBFILTER()

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Caracter</b>	Retorna a expressão do filtro ativo na área de trabalho atual. Caso não exista filtro ativo retorna "" (String vazia).
-----------------	--

**Exemplo:**

```
USE Cliente INDEX Ind1 NEW  
SET FILTER TO Nome > "Jose"  
DBFILTER() // retorna: Nome > "Jose"  
SET FILTER TO Num < 1000  
DBFILTER() // retorna: Num < 1000
```



*Anotações*

---

---

---

---

## **DBGOTO()**

Move o cursor da área de trabalho ativa para o record number (recno) especificado, realizando um posicionamento direto, sem a necessidade uma busca (seek) prévio.

- Sintaxe: DbGoto(nRecno)**
- Parâmetros**

<b>nRecno</b>	Record number do registro a ser posicionado.
---------------	--

### **Exemplo:**

```
DbSelectArea("SA1")
DbGoto(100) // Posiciona no registro 100
```

```
IF !EOF() // Se a área de trabalho não estiver em final de arquivo
    MsgInfo("Você está no cliente:" + A1_NOME)
ENDIF
```

## **DBGOTOP()**

Move o cursor da área de trabalho ativa para o primeiro registro lógico.

- Sintaxe: DbGoTop()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
nCount := 0 // Variável para verificar quantos registros há no intervalo
DbSelectArea("SA1")
DbSetOrder(1) // A1_FILIAL + A1_COD + A1_LOJA
DbGoTop()

While !BOF() // Enquanto não for o início do arquivo
    nCount++ // Incrementa a variável de controle de registros no intervalo
    DbSkip(-1)
End

MsgInfo("Existem :" + STRZERO(nCount,6) + " registros no intervalo").

// Retorno esperado :000001, pois o DbGoTop posiciona no primeiro registro.
```

## **DBGOBOTTON()**

Move o cursor da área de trabalho ativa para o último registro lógico.

- Sintaxe: DbGoButton()**
- Parâmetros**

**Nenhum**

.

### **Exemplo:**

```

nCount := 0 // Variável para verificar quantos registros há no intervalo
DbSelectArea("SA1")
DbSetOrder(1) // A1_FILIAL + A1_COD + A1_LOJA
DbGoButton()

While !EOF() // Enquanto não for o início do arquivo
    nCount++ // Incrementa a variável de controle de registros no intervalo
    DbSkip(1)
End

MsgInfo("Existem :" + STRZERO(nCount,6) + " registros no intervalo").

// Retorno esperado :000001, pois o DbGoButton posiciona no último registro.

```

## **DBINFO()**

DBINFO() é utilizada para obter informações sobre a tabela corrente. O tipo de informação (primeiro argumento) é escolhido de acordo com as constantes abaixo:

- Tabela A : Constantes utilizadas na parametrização da função**

<b>Constante</b>	<b>Descrição</b>	<b>Retorno</b>
<b>DBI_GETRECSIZE</b>	Tamanho do registro em número de bytes similar a RECSIZE.	Numérico
<b>DBI_TABLEEXT</b>	Extensão do arquivo da tabela corrente.	Caracter
<b>DBI_FULLPATH</b>	Nome da tabela corrente com caminho completo.	Caracter
<b>DBI_BOF</b>	Se está posicionada no início da tabela similar a BOF	Lógico
<b>DBI_EOF</b>	Se está posicionada no final da tabela similar a EOF	Lógico
<b>DBI_FOUND</b>	Se a tabela está posicionada após uma pesquisa similar a FOUND	Lógico
<b>DBI_FCOUNT</b>	Número de campos na estrutura da tabela corrente similar a FCOUNT	Numérico
<b>DBI_ALIAS</b>	Nome do Alias da área de trabalho corrente similar a ALIAS	Caracter
<b>DBI_LASTUPDATE</b>	Data da última modificação similar a LUPDATE	Data

- Sintaxe:** DBINFO(<nINFOTIPO>)

- Parâmetros:**

<b>nINFOTIPO</b>	Tipo de informação a ser verificada.
------------------	--------------------------------------

- Retorno:**

<b>Indefinido</b>	Informação da Tabela Informação requisitada pelo usuário (o tipo depende da informação requisitada). Se não houver tabela corrente retorna NIL.
-------------------	---

**Exemplo:**

```
USE Clientes NEW
DBINFO(DBI_FULLPATH) // Retorno: C:\Teste\Clientes.dbf
DBINFO(DBI_FCOUNT) // Retorno: 12
DBGOTOP()
DBINFO(DBI_BOF) // Retorno: .F.
DBSKIP(-1)
DBINFO(DBI_BOF) // Retorno: .T.
```

### **DBNICKINDEXKEY()**

Função que retorna o "IndexKey", ou seja, a expressão de índice da ordem especificada pelo NickName. Se não existe índice com o nickname, retorna uma string vazia.

- Sintaxe:** DBNICKINDEXKEY(<cNick>)

- Parâmetros:**

<b>cNick</b>	Indica o "NickName" da ordem de índice.
--------------	---

- Retorno:**

<b>Caracter</b>	Expressão do índice identificado pelo "NickName".
-----------------	---



Anotações

---

---

---

---

---

## **DBORDERINFO()**

A função DBORDERINFO() é utilizada para obter informações sobre determinada ordem. A especificação da ordem pode ser realizada através de seu nome ou sua posição dentro da lista de ordens, mas se ela não for especificada serão obtidas informações da ordem corrente. O tipo de informação (primeiro argumento) é escolhido de acordo com as constantes abaixo:

- Tabela A : Constantes utilizadas na parametrização da função**

<b>Constante</b>	<b>Descrição</b>	<b>Retorno</b>
<b>DBOI_BAGNAME</b>	Nome do arquivo de índice ao qual a ordem pertence.	Caracter
<b>DBOI_FULLPATH</b>	do arquivo de índice (com seu diretório) ao qual a ordem pertence.	Caracter
<b>DBOI_ORDERCOUNT</b>	Número de ordens existentes no arquivo de índice especificado.	Caracter

- Sintaxe: DBORDERINFO(<nINFOTIPO>)**

- Parâmetros:**

<b>nINFOTIPO</b>	Nome do arquivo de índice.
------------------	----------------------------

- Retorno:**

<b>Caracter</b>	Retorna a informação da Ordem requisitada pelo usuário (pode ser de tipo numérico se for número de ordens no índice, tipo caracter se for nome do arquivo de índice). Caso não exista ordem corrente ou a posição da ordem especificada está inválida retorna NIL.
-----------------	--

### **Exemplo:**

DBORDERINFO(DBOI_BAGNAME) // retorna: Ind DBORDERINFO(DBOI_FULLPATH) // retorna: C:\AP6\Teste\Ind.cdx
--



*Anotações*

---

---

---

---

## **DBORDERNICKNAME()**

A função DBORDERNICKNAME() é utilizada para selecionar a ordem ativa através de seu apelido. Esta ordem é a responsável pela seqüência lógica dos registros da tabela corrente.

- Sintaxe:** **DBORDERNICKNAME(<cAPELIDO>)**

- Parâmetros:**

<b>cAPELIDO</b>	Nome do apelido da ordem a ser setada.
-----------------	--

- Retorno:**

<b>Lógico</b>	Retorna Falso se não conseguiu tornar a ordem ativa. Principais erros: Não existe tabela ativa ou não foi encontrada a ordem com o apelido. Retorna Verdadeiro se a ordem foi setada com sucesso.
---------------	---

### **Exemplo:**

```
USE Cliente NEW
SET INDEX TO Nome, Idade

IF !DBORDERNICKNAME("IndNome")
    Messagebox("Registro não encontrado","Erro", 0)
ENDIF
```

## **DBPACK()**

A função DBPACK() remove fisicamente todos os registros com marca de excluído da tabela.

- Sintaxe:** **\_\_DBPACK()**
- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
USE Clientes NEW
DBGOTO(100)
DBDELETE()
DBGOTO(105)
DBDELETE()
DBGOTO(110)
DBDELETE()

// Se a exclusão for confirmada:
__DBPACK()
```

## **DBRECALL()**

A função DBRECALL() é utilizada para retirar a marca de registro deletado do registro atual. Para ser executada o registro atual deve estar bloqueado ou a tabela deve estar aberta em modo exclusivo. Se o registro atual não estiver deletado, esta função não faz nada. Ela é o oposto da função DBDELETE() que marca o registro atual como deletado.

- Sintaxe: DBRECALL()**

- Parâmetros:**

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

- Retorno:**

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

### **Exemplo 01: Desfazendo a deleção do registro posicionado do alias corrente**

```
USE Cliente
DBGOTO(100)
DBDELETE()
DELETED() // Retorna: .T.
DBRECALL()
DELETED() // Retorna: .F.
```

### **Exemplo 02: Desfazendo as deleções do alias corrente**

```
USE Cliente
DBGOTOP()
WHILE !EOF()
    DBRECALL()
    DBSKIP()
ENDDO
```

## **DBRECORDINFO()**

A função DBRECORDINFO() é utilizada para obter informações sobre o registro especificado pelo segundo argumento (recno) da tabela corrente, se esta informação for omitida será verificado o registro corrente. O tipo de informação (primeiro argumento) é escolhido de acordo com as constantes abaixo:

- Tabela A : Constantes utilizadas na parametrização da função**

Constante	Descrição	Retorno
<b>DBRI_DELETED</b>	Estado de deletado similar a DELETED	Lógico
<b>DBRI_RECSIZE</b>	Tamanho do registro similar a RECSIZE	Numérico
<b>DBRI_UPDATED</b>	Verifica se o registro foi alterado e ainda não foi atualizado fisicamente similar a UPDATED	Lógico

- Sintaxe:** DBRECORDINFO ( < nINFOTIPO > , [ nREGISTRO ] ) --> xINFO

**Parâmetros:**

nINFOTIPO	Tipo de informação a ser verificada.
nREGISTRO	Número do registro a ser verificado.

**Retorno:**

<b>Indefinido</b>	Não há tabela corrente ou registro inválido. Informação do Registro. Informação requisitada pelo usuário (o tipo depende da informação requisitada).
-------------------	--

**Exemplo:**

```
USE Clientes NEW
DBGOTO(100)
DBRECORDINFO(DBRI_DELETED) // Retorno: .F.
DBDELETE()
DBRECORDINFO(DBRI_DELETED) // Retorno: .F.
DBRECALL()
DBRECORDINFO(DBRI_RECSIZE) // Retorno: 230
NOME := "JOAO"
DBGOTO(200)
DBRECORDINFO(DBRI_UPDATED) // Retorno: .F.
DBRECORDINFO(DBRI_UPDATED,100) // Retorno: .T.
```

**DBREINDEX()**

A função DBREINDEX() reconstrói todos os índices da área de trabalho corrente e posiciona as tabelas no primeiro registro lógico.

- Sintaxe: DBREINDEX()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
USE Clientes NEW
DBSETINDEX("IndNome")
DBREINDEX()
```

## **DBRLOCK()**

---

A função DBRLOCK() é utilizada quando se tem uma tabela aberta e compartilhada e se deseja bloquear um registro para que outros usuários não possam alterá-lo. Se a tabela já está aberta em modo exclusivo, a função não altera seu estado. O usuário pode escolher o registro a ser bloqueado através do parâmetro (recno), mas se este for omitido será bloqueado o registro corrente como na função RLOCK(). Esta função é o oposto à DBRUNLOCK, que libera registros bloqueados.

- Sintaxe:** **DBRLOCK([nREGISTRO])**

- Parâmetros:**

nREGISTRO	Número do registro a ser bloqueado.
-----------	-------------------------------------

- Retorno:**

<b>Lógico</b>	Retorna Falso se não conseguiu bloquear o registro. Principal motivo: o registro já foi bloqueado por outro usuário.  Retorna Verdadeiro se o registro foi bloqueado com sucesso.
---------------	---

### **Exemplo:**

```
DBUSEAREA( .T.,"dbfcxdxads", "\dadosadv609\sa1990.dbf","SSS",.T., .F. )
DBGOTO(100)
DBRLOCK() // Bloqueia o registro atual (100)
DBRLOCK(110) // Bloqueia o registro de número 110
```

---

## **DBRLOCKLIST()**

A função DBRLOCKLIST() é utilizada para verificar quais registros estão locados na tabela corrente. Para tanto, é retornada uma tabela unidimensional com os números dos registros.

- Sintaxe:** **DBRLOCKLIST()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Array</b>	Retorna NIL se não existe tabela corrente ou não existe nenhum registro locado. Retorna a lista com os recnos dos registros locados na tabela corrente.
--------------	---

**Exemplo:**

```
DBUSEAREA( .T.,"dbfcxdads", "\dadosadv609\sa1990.dbf","SSS",.T., .F. )
DBGOTOP()
DBRLOCK() // Bloqueia o primeiro registro
DBRLOCK(110) // Bloqueia o registro de número 110
DBRLOCK(100) // Bloqueia o registro de número 100
DBRLOCKLIST() // Retorna: {1,100,110}
```

**DBRUNLOCK()**

A função DBRUNLOCK() é utilizada para liberar determinado registro bloqueado. O usuário pode escolher o registro a ser desbloqueado através do parâmetro (Recno), mas se este for omitido será desbloqueado o registro corrente como na função DBUNLOCK(). Esta função é o oposto à DBRLOCK, que bloqueia os registros.

- Sintaxe: DBRUNLOCK([nREGISTRO])**

- Parâmetros:**

<b>nREGISTRO</b>	Número do registro a ser desbloqueado.
------------------	--

- Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
DBUSEAREA( .T.,"dbfcxdads", "\dadosadv609\sa1990.dbf","SSS",.T., .F. )
DBGOTO(100)
DBRUNLOCK() //Desbloqueia o registro atual (100)
DBRUNLOCK(110) // Desbloqueia o registro de número 110
```

**DBSETDRIVER()**

A função DBSETDRIVER() pode ser utilizada apenas para verificar qual o RDD que está definido como padrão quando for omitido seu parâmetro. Ela também pode ser utilizada para especificar outro RDD como padrão, especificando-o através do parâmetro.

- Sintaxe: DBSETDRIVER([cNOVORDD])**

- Parâmetros:**

<b>cNOVORDD</b>	Novo nome do RDD a ser definido como padrão.
-----------------	--

- Retorno:**

<b>Caracter</b>	Nome do RDD padrão corrente.
-----------------	------------------------------

**Exemplo:**

```
DBSETDRIVER("CTREECDX") // Retorna: DBFCDX  
DBSETDRIVER() // Retorna: CTREECDX
```



*Importante*

Note que ao utilizar a função DBSETDRIVER para redefinir o driver corrente, o retorno da função não será o driver definido nos parâmetros, mas o driver que estava em uso antes da atualização.

## **DBSETINDEX()**

A função DBSETINDEX() é utilizada para acrescentar uma ou mais ordens de determinado índice na lista de ordens ativas da área de trabalho. Quando o arquivo de índice possui apenas uma ordem, a mesma é acrescentada à lista e torna-se ativa. Quando o índice possui mais de uma ordem, todas são acrescentadas à lista e a primeira torna-se ativa.



*Importante*

Para utilizar os arquivos de extensão padrão do RDD este dado deve ser especificado.

- Sintaxe: DBSETINDEX(<@cARQINDICE>)**
- Parâmetros:**

<b>cARQINDICE</b>	Nome do arquivo de índice, com ou sem diretório.
-------------------	--

- Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
USE Cliente NEW  
DBSETINDEX("Ind1")  
DBSETINDEX("\teste\Ind2.cdx")
```



*Anotações*

---

---

---

---

---

## **DBSETNICKNAME()**

A função DBSETNICKNAME() é utilizada para colocar um apelido em determinada ordem especificada pelo primeiro parâmetro. Caso seja omitido o nome do apelido a ser dado, a função apenas verifica o apelido corrente.

- Sintaxe:** **DBSETNICKNAME(<cINDICE>, [cAPELIDO])**

- Parâmetros:**

cINDICE	Nome da ordem que deve receber o apelido.
cAPELIDO	Nome do apelido da ordem a ser setada.

- Retorno:**

Caracter	Retorna "" (String vazia) se não conseguiu encontrar a ordem especificada, não conseguiu setar o apelido ou não havia apelido. Retorna o apelido corrente.
----------	--

### **Exemplo:**

```
USE Cliente NEW
DBSETNICKNAME("IndNome") // retorna: ""
DBSETNICKNAME("IndNome","NOME") // retorna: ""
DBSETNICKNAME("IndNome") // retorna: "NOME"
```

## **DBSELECTAREA()**

Define a área de trabalho especificada com sendo a área ativa. Todas as operações subsequentes que fizerem referência a uma área de trabalho a utilização, a menos que a área desejada seja informada explicitamente.

- Sintaxe:** **DbSelectArea(nArea | cArea)**
- Parâmetros**

nArea	Valor numérico que representa a área desejada, em função de todas as áreas já abertas pela aplicação, que pode ser utilizado ao invés do nome da área.
cArea	Nome de referência da área de trabalho a ser selecionada.

### **Exemplo 01: DbselectArea(nArea)**

```
nArea := Select("SA1") // → 10 (proposto)

DbSelectArea(nArea) // De acordo com o retorno do comando Select()

ALERT("Nome do cliente: "+A1_NOME) // Como o SA1 é o alias selecionado, os comandos
// a partir da seleção do alias compreendem que ele
// está implícito na expressão, o que causa o mesmo
// efeito de SA1->A1_NOME
```

### **Exemplo 02: DbselectArea(cArea)**

```
DbSelectArea("SA1") // Especificação direta do alias que deseja-se selecionar  
  
ALERT("Nome do cliente: "+A1_NOME) // Como o SA1 é o alias selecionado, os comandos  
// a partir da seleção do alias compreendem que ele  
// está implícito na expressão, o que causa o mesmo  
// efeito de SA1->A1_NOME
```

### **DBSETORDER()**

Define qual índice será utilizada pela área de trabalho ativa, ou seja, pela área previamente selecionada através do comando DbSelectArea(). As ordens disponíveis no ambiente Protheus são aquelas definidas no SINDEX /SIX, ou as ordens disponibilizadas por meio de índices temporários.

- Sintaxe: DbSetOrder(nOrdem)**
- Parâmetros**

<b>nOrdem</b>	Número de referência da ordem que deseja ser definida como ordem ativa para a área de trabalho.
---------------	---

#### **Exemplo:**

```
DbSelectArea("SA1")  
DbSetOrder(1) // De acordo com o arquivo SIX -> A1_FILIAL+A1_COD+A1_LOJA
```

### **DBORDERNICKNAME()**

Define qual índice criado pelo usuário seja utilizado. O usuário pode incluir os seus próprios índices e no momento da inclusão deve criar o NICKNAME para o mesmo.

- Sintaxe: DbOrderNickName(NickName)**
- Parâmetros**

<b>NickName</b>	NickName atribuído ao índice criado pelo usuário
-----------------	--

#### **Exemplo:**

```
DbSelectArea("SA1")  
DbOrderNickName("Tipo") // De acordo com o arquivo SIX -> A1_FILIAL+A1_TIPO  
NickName: Tipo
```

## **DBSEEK() E MSSEEK()**

**DbSeek():** Permite posicionar o cursor da área de trabalho ativo no registro com as informações especificadas na chave de busca, fornecendo um retorno lógico indicando se o posicionamento foi efetuado com sucesso, ou seja, se a informação especificada na chave de busca foi localizada na área de trabalho.

- Sintaxe:** DbSeek(**cChave, ISoftSeek, ILast**)
- Parâmetros**

<b>cChave</b>	Dados do registro que deseja-se localizar, de acordo com a ordem de busca previamente especificada pelo comando DbSetOrder(), ou seja, de acordo com o índice ativo no momento para a área de trabalho.
<b>ISoftSeek</b>	Define se o cursor ficará posicionado no próximo registro válido, em relação a chave de busca especificada, ou em final de arquivo, caso não seja encontrada exatamente a informação da chave. Padrão → .F.
<b>ILast</b>	Define se o cursor será posicionado no primeiro ou no último registro de um intervalo com as mesmas informações especificadas na chave. Padrão → .F.

### **Exemplo 01 – Busca exata**

```
DbSelectArea("SA1")
DbSetOrder(1) // acordo com o arquivo SIX -> A1_FILIAL+A1_COD+A1_LOJA

IF DbSeek("01" + "000001" + "02" ) // Filial: 01, Código: 000001, Loja: 02
    MsgInfo("Cliente localizado", "Consulta por cliente")
Else
    MsgAlert("Cliente não encontrado", "Consulta por cliente")
Endif
```

### **Exemplo 02 – Busca aproximada**

```
DbSelectArea("SA1")
DbSetOrder(1) // acordo com o arquivo SIX -> A1_FILIAL+A1_COD+A1_LOJA

DbSeek("01" + "000001" + "02", .T. ) // Filial: 01, Código: 000001, Loja: 02

// Exibe os dados do cliente localizado, o qual pode não ser o especificado na chave:

MsgInfo("Dados do cliente localizado: "+CRLF +
        "Filial:" + A1_FILIAL + CRLF +
        "Código:" + A1_COD + CRLF +
        "Loja:" + A1_LOJA + CRLF +
        "Nome:" + A1_NOME + CRLF, "Consulta por cliente")
```

**MsSeek():** Função desenvolvida pela área de Tecnologia da Microsiga, a qual possui as mesmas funcionalidades básicas da função DbSeek(), com a vantagem de não necessitar acessar novamente a base de dados para localizar uma informação já utilizada pela thread (conexão) ativa.

Desta forma, a thread mantém em memória os dados necessários para reposicionar os registros já localizados através do comando DbSeek (no caso o Recno()) de forma que a aplicação pode simplesmente efetuar o posicionamento sem executar novamente a busca.

A diferença entre o DbSeek() e o MsSeek() é notada em aplicações com grande volume de posicionamentos, como relatórios, que necessitam referenciar diversas vezes o mesmo registro durante uma execução.

## **DBSKIP()**

Move o cursor do registro posicionado para o próximo (ou anterior dependendo do parâmetro), em função da ordem ativa para a área de trabalho.

- Sintaxe:** **DbSkip(nRegistros)**
- Parâmetros**

<b>nRegistros</b>	Define em quantos registros o cursor será deslocado. Padrão → 1
-------------------	---

### **Exemplo 01 – Avançando registros**

```
DbSelectArea("SA1")
DbSetOrder(2) // A1_FILIAL + A1_NOME
DbGotop() // Posiciona o cursor no início da área de trabalho ativa

While !EOF() // Enquanto o cursor da área de trabalho ativa não indicar fim de arquivo
    MsgInfo("Você está no cliente:" + A1_NOME)
    DbSkip()
End
```

### **Exemplo 02 – Retrocedendo registros**

```
DbSelectArea("SA1")
DbSetOrder(2) // A1_FILIAL + A1_NOME
DbGoBottom() // Posiciona o cursor no final da área de trabalho ativa

While !BOF() // Enquanto o cursor da área de trabalho ativa não indicar início de arquivo
    MsgInfo("Você está no cliente:" + A1_NOME)
    DbSkip(-1)
End
```

## **DBSETFILTER()**

Define um filtro para a área de trabalho ativa, o qual pode ser descrito na forma de um bloco de código ou através de uma expressão simples.

- Sintaxe:** **DbSetFilter(bCondicao, cCondicao)**
- Parâmetros**

<b>bCondicao</b>	Bloco de expressa a condição de filtro em forma executável
<b>cCondicao</b>	Expressão de filtro simples na forma de string

### **Exemplo 01 – Filtro com bloco de código**

```
bCondicao := {|| A1_COD >= "000001" .AND. A1_COD <= "001000"}  
DbSelectArea("SA1")  
DbSetOrder(1)  
DbSetFilter(bCondicao)  
DbGoBotton()  
  
While !EOF()  
    MsgInfo("Você está no cliente:" + A1_COD)  
    DbSkip()  
End  
  
// O último cliente visualizado deve ter o código menor do que "001000".
```

### **Exemplo 02 – Filtro com expressão simples**

```
cCondicao := "A1_COD >= '000001' .AND. A1_COD <= '001000'"  
DbSelectArea("SA1")  
DbSetOrder(1)  
DbSetFilter(,cCondicao)  
DbGoBotton()  
  
While !EOF()  
    MsgInfo("Você está no cliente:" + A1_COD)  
    DbSkip()  
End  
  
// O último cliente visualizado deve ter o código menor do que "001000".
```

## **DBSTRUCT()**

---

Retorna um array contendo a estrutura da área de trabalho (alias) ativo. A estrutura será um array bidimensional conforme abaixo:

<b>ID*</b>	<b>Nome campo</b>	<b>Tipo campo</b>	<b>Tamanho</b>	<b>Decimais</b>
------------	-------------------	-------------------	----------------	-----------------

\*Índice do array

- Sintaxe: DbStruct()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
cCampos := ""  
DbSelectArea("SA1")  
aStructSA1 := DbStruct()  
  
FOR nX := 1 to Len(aStructSA1)  
    cCampos += aStructSA1[nX][1] + "/"  
  
NEXT nX  
  
ALERT(cCampos)
```

---

## **DBUNLOCK()**

---

A função DBUNCLOK() retira os bloqueios dos registros e do arquivo da tabela corrente.

- Sintaxe: DBUNLOCK()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **DBUNLOCKALL()**

---

A função DBUNLOCKALL() Retira os bloqueios de todos os registros e dos arquivos de todas as tabelas abertas. Esta função é utilizada para liberar todos os registros bloqueados e é equivalente a executar DBUNLOCK para todas as tabelas da área de trabalho.

- Sintaxe: DBUNLOCKALL()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **DBUSEAREA()**

---

Define um arquivo de base de dados como uma área de trabalho disponível na aplicação.

- Sintaxe: DbUseArea(**INovo**, **cDriver**, **cArquivo**, **cAlias**, **IComparilhado**,; **ISoLeitura**)**

- Parâmetros**

<b>INovo</b>	Parâmetro opcional que permite que se caso o cAlias especificado já esteja em uso, ele seja fechado antes da abertura do arquivo da base de dados.
<b>cDriver</b>	Driver que permita a aplicação manipular o arquivo de base de dados especificado. A aplicação ERP possui a variável __LOCALDRIVER definida a partir das configurações do .ini do server da aplicação. Algumas chaves válidas: "DBFCDX", "CTREECDX", "DBFCDXAX", "TOPCONN".
<b>cArquivo</b>	Nome do arquivo de base de dados que será aberto com o alias especificado.
<b>cAlias</b>	Alias para referência do arquivos de base de dados pela aplicação.
<b>IComparilhado</b>	Se o arquivo poderá ser utilizado por outras conexões.
<b>ISoLeitura</b>	Se o arquivo poderá ser alterado pela conexão ativa.

### **Exemplo:**

```
DbUserArea(.T., "DBFCDX", "\SA1010.DBF", "SA1DBF", .T., .F.)  
DbSelectArea("SA1DBF")  
MsgInfo("A tabela SA1010.DBF possui:" + STRZERO(RecCount(),6) + " registros.")  
DbCloseArea()
```

## **DELETED()**

---

A função DELETED() Verifica se o registro está com marca de excluído. Quando o registro é excluído, permanece fisicamente na tabela, mas fica marcado como excluído. Esta função verifica este estado. Se nenhuma área está selecionada, retorna .F.. Quando é executada a função DBPACK() todos os registros marcados como deletados são apagados fisicamente. A função DBRECALL() retira todas as marcas.

- Sintaxe: DELETED()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
USE "\DADOSADV\AA1990.DBF" SHARED NEW  
DBGOTO(100)  
IF DELETED()  
Messagebox("O registro atual foi deletado","Erro", 0)  
ENDIF
```

## **FCOUNT()**

---

A função FCOUNT() avalia a quantidade de campos existentes na estrutura do arquivo ativo como área de trabalho.

- Sintaxe: FCOUNT()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Numérico</b>	Quantidade de campos existentes na estrutura da área de trabalho ativa.
-----------------	---

### **Exemplo:**

```
DbSelectArea("SA1")  
nFields := FCOUNT()  
  
IF nFields > 0  
    MSGINFO("A estrutura da tabela contém :+CvalToChar(nFields)+\"campos.\")  
ENDIF
```

## **FOUND()**

A função FOUND() recupera o resultado de sucesso referente a última operação de busca efetuada pelo processamento corrente.

**Sintaxe: FOUND()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Lógico</b>	Indica se a última operação de busca realizada pelo processamento corrente obteve sucesso (.T.) ou não (.F.).
---------------	---

**Exemplo:**

```
Pergunte(cPerg,,.T.)
DbSelectArea("SA1")
DbSetOrder(1)
DbSeek(xFilial("SA1")+MVPAR01)

IF Found()
    MSGINFO("Cliente encontrado")
ELSE
    MSGALERT("Dados não encontrados")
ENDIF
```

## **INDEXKEY()**

A função INDEXKEY() determina a expressão da chave de um índice especificado na área de trabalho corrente, e o retorna na forma de uma cadeia de caracteres, sendo normalmente utilizada na área de trabalho correntemente selecionada.

**Sintaxe: INDEXKEY()**

**Parâmetros:**

<b>nOrdem</b>	Ordem do índice na lista de índices abertos pelo comando USE...INDEX ou SET INDEX TO na área de trabalho corrente. O valor default zero especifica o índice corrente, independentemente de sua posição real na lista.
---------------	---

**Retorno:**

<b>Caracter</b>	Expressão da chave do índice especificado na forma de uma cadeia de caracteres. Caso não haja um índice correspondente, INDEXKEY() retorna uma cadeia de caracteres vazia ("").
-----------------	---

**Exemplo:**

```
cExpressao := SA1->(IndexKey())
```

## **INDEXORD()**

A função INDEXORD() verifica a posição do índice corrente na lista de índices do respectivo alias.

**Sintaxe: INDEXORD()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Numérico</b>	Posição do índice corrente na lista de índices da tabela. Retorna 0 se não existe índice aberto na tabela corrente.
-----------------	---

### **Exemplo:**

```
USE Cliente NEW
SET INDEX TO Nome, End, Cep
nOrd:=INDEXORD() // Return: 1 - é o primeiro índice da lista
```

## **LUPDATE()**

A função LUPDATE() verifica qual a data da última modificação e fechamento da tabela corrente, sendo que caso não exista tabela corrente é retornada uma data em branco.

**Sintaxe: LUPDATE()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Data</b>	Retorna um valor do tipo Data , indicando a data da ultima modificação e fechamento da Tabela. Caso não haja tabela selecionada na área de trabalho atual , a função retornará uma data vazia (ctod("")) .
-------------	--

### **Exemplo:**

```
// Mostra a data da última modificação da tabela corrente,
dModificacao := LUpdate()
IF (EMPTY(dModificacao))
    CONOUT("Não há tabela corrente")
ELSE
    CONOUT(("Data da ultima modificacao : " + DTOS(dModificacao)))
ENDIF
```

## **MSAPPEND()**

A função MsAppend() adiciona registros de um arquivo para outro, respeitando a estrutura das tabelas.

- Sintaxe: MSAPPEND( [cArqDest], cArqOrig )**

- Parâmetros:**

<b>cArqDest</b>	Se o RDD corrente for DBFCDX os registros serão adicionados na área selecionada, caso contrário o arquivo destino terá que ser informado.
<b>cArqOrig</b>	Nome do arquivo origem contendo os registros a serem adicionados.

- Retorno:**

<b>Lógico</b>	Se a operação for realizada com sucesso o função retornará verdadeiro (.T.).
---------------	--

**Exemplo:**

```
dbSelectArea('XXX')
MsAppend(,'ARQ00001')
```

## **MSUNLOCK()**

Libera o travamento (lock) do registro posicionado confirmando as atualizações efetuadas neste registro.

- Sintaxe: MsUnLock()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
DbSelectArea("SA1")
DbSetOrder(1) // A1_FILIAL + A1_COD + A1_LOJA
DbSeek("01" + "900001" + "01") // Busca exata

IF Found() // Avalia o retorno do último DbSeek realizado
  RecLock("SA1",.F.)
  SA1->A1_NOME := "CLIENTE CURSO ADVPL BÁSICO"
  SA1->A1_NREDUZ := "ADVPL BÁSICO"
  MsUnLock() // Confirma e finaliza a operação
ENDIF
```

## **ORDBAGEXT()**

A função ORDBAGEXT é utilizada no gerenciamento de índices para os arquivos de dados do sistema, permitindo avaliar qual a extensão deste índice atualmente em uso, de acordo com a RDD ativa.

**Sintaxe: ORDBAGEXT()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>cBagExt</b>	Extensão do arquivo dos arquivos de índices em uso pelo sistema, determinado pela RDD ativa.
----------------	--

**Exemplo:**

```
cArqTRB := CriaTrab(aStruTRB,.T.)
// RDD UTILIZADA: "DBFCDXADS"
DbUseArea(.T., "DBFCDXADS", cArqTRB, "TRBSA1", .F., .F.)

DbSelectArea("TRBSA1")
cArqInd := CriaTrab(Nil,.F.)
IndRegua("TRBSA1",cArqInd,cChaveInd,,,"Selecionando registros ...")
#IFNDEF TOP
    DbSetIndex(cArqInd+OrdBagExt())
    // RETORNO: ".CDX"
#ENDIF
DbSetOrder(1)
```

## **ORDKEY()**

A função ORDKEY() verifica qual é a expressão de chave de determinada ordem. Caso não sejam especificados os parâmetros de identificação da ordem, é verificada a ordem corrente. Para evitar conflito, no caso de haver mais de uma ordem com o mesmo nome, pode-se passar o parâmetro com o nome do índice ao qual a ordem pertence.

A ordem passada no primeiro parâmetro pode ser especificada através da sua posição na lista de ordens ativas (através do ORDLISTADD) ou através do nome dado à ordem. A função verifica automaticamente se o parâmetro é numérico ou caractere.

**Sintaxe: ORDKEY([cOrdem | nPosicao] , [cArqIndice])**

**Parâmetros:**

<b>cOrdem</b>	Há duas opções para o primeiro parâmetro: cNome: tipo caractere, contém nome do índice. nPosicao: tipo numérico, indica ordem do índice.
<b>cArqIndice</b>	Nome do arquivo de índice.

- Retorno:**

<b>Caracter</b>	Expressão de chave da ordem ativa ou especificada pelos parâmetros. Cadeia vazia indica que não existe ordem corrente.
-----------------	---

**Exemplo:**

```
USE Cliente NEW
INDEX ON Nome+Cod TO Ind1 FOR Nome+Cod > 'AZZZZZZZ'
ORDKEY('Ind1')
// Retorna: Nome+Cod
```

## **RECLKLOCK()**

Efetua o travamento do registro posicionado na área de trabalho ativa, permitindo a inclusão ou alteração das informações do mesmo.

- Sintaxe: RecLock(cAlias,!Inclui)**  
 **Parâmetros**

<b>cAlias</b>	Alias que identifica a área de trabalho que será manipulada.
<b>!Inclui</b>	Define se a operação será uma inclusão (.T.) ou uma alteração (.F.)

**Exemplo 01 - Inclusão**

```
DbSelectArea("SA1")
RecLock("SA1",.T.)
SA1->A1_FILIAL := xFilial("SA1") // Retorna a filial de acordo com as configurações do ERP
SA1->A1_COD := "900001"
SA1->A1_LOJA := "01"
MsUnLock() // Confirma e finaliza a operação
```

**Exemplo 02 - Alteração**

```
DbSelectArea("SA1")
DbSetOrder(1) // A1_FILIAL + A1_COD + A1_LOJA
DbSeek("01" + "900001" + "01") // Busca exata

IF Found() // Avalia o retorno do último DbSeek realizado
  RecLock("SA1",.F.)
  SA1->A1_NOME := "CLIENTE CURSO ADVPL BÁSICO"
  SA1->A1_NREDUZ := "ADVPL BÁSICO"
  MsUnLock() // Confirma e finaliza a operação
ENDIF
```



Dica

A linguagem ADVPL possui variações da função RecLock(), as quais são:

- RLOCK()
- DBRLOCK()

A sintaxe e a descrição destas funções estão disponíveis no Guia de Referência Rápido ao final deste material.

A linguagem ADVPL possui variações da função MsUnlock(), as quais são:



Dica

- UNLOCK()
- DBUNLOCK()
- DBUNLOCKALL()

A sintaxe e a descrição destas funções estão disponíveis no Guia de Referência Rápido ao final deste material.

## RECNO()

A função RECNO() retorna o número do registro atualmente posiconado na área de trabalho ativa.

**Sintaxe: RECNO()**

**Parâmetros:**

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

**Retorno:**

<b>nRecno</b>	Identificador numérico do registro atualmente posicionando na área de trabalho ativa.
---------------	---

**Exemplo:**

```
DbSelectArea("SA1")
DbGoto(100) // Posiciona no registro de recno 100.
MSGINFO("Registro posicionado:" + cValToChar(RECNO()))
```

## SELECT()

A função SELECT() determina o número da área de trabalho de um alias. O número retornado pode variar de zero a 250. Se <cAlias> não for especificado, é retornado o número da área de trabalho corrente. Caso <cAlias> seja especificado e o alias não existir, SELECT() retorna zero.

**Sintaxe: SELECT([cAlias])**

**Parâmetros:**

<b>cAlias</b>	Nome da área de trabalho a ser verificada.
---------------	--

**Retorno:**

<b>Numérico</b>	Área de trabalho do alias especificado na forma de um valor numérico inteiro.
-----------------	---

**Exemplo:**

```
nArea := Select("SA1")
```

```
ALERT("Referência do alias SA1: "+STRZERO(nArea,3)) // → 10 (proposto)
```

**SET FILTER TO**

O comando SET FILTER TO define uma condição de filtro que será aplicada a área de trabalho ativa.



*Importante*

Recomenda-se o uso da função DbSetFilter() em substituição ao comando SET FILTER TO

**Sintaxe: SET FILTER TO cCondicao**

**Parâmetros:**

**cCondicao**

Expressão que será avaliada pela SET FILTER, definindo os registros que ficarão disponíveis na área de trabalho ativa.  
Esta expressão obrigatoriamente deve ter um retorno lógico.

**Retorno:**

**Nenhum**

.



*Dica*

O uso da sintaxe SET FILTER TO desativa o filtro na área de trabalho corrente.

**Exemplo:**

```
USE Employee INDEX Name NEW
```

```
SET FILTER TO Age > 50
```

```
LIST LastName, FirstName, Age, Phone
```

```
SET FILTER TO
```

## **SOFTLOCK()**

Permite a reserva do registro posicionado na área de trabalho ativa de forma que outras operações, com exceção da atual, não possam atualizar este registro. Difere da função RecLock() pois não gera uma obrigação de atualização, e pode ser sucedido por ele.

Na aplicação ERP Protheus, o SoftLock() é utilizado nos browses, antes da confirmação da operação de alteração e exclusão, pois neste momento a mesma ainda não foi efetivada, mas outras conexões não podem acessar aquele registro pois o mesmo está em manutenção, o que implementa da integridade da informação.

**Sintaxe: SoftLock(cAlias)**

**Parâmetros**

<b>cAlias</b>	Alias de referência da área de trabalho ativa, para o qual o registro posicionado será travado.
---------------	---

**Exemplo:**

```
cChave := GetCliente() // Função ilustrativa que retorna os dados de busca de um cliente  
  
DbSelectArea("SA1")  
DbSetOrder(1)  
DbSeek(cChave)  
  
IF Found()  
    SoftLock() // Reserva o registro localizado  
    IConfirma := AlteraSA1() // Função ilustrativa que exibe os dados do registro  
    // posicionado e permite a alteração dos mesmos.  
  
    IF IConfirma  
        RecLock("SA1",.F.)  
        GravaSA1() // Função ilustrativa que altera os dados conforme a AlertaSA1()  
        MsUnLock() // Liberado o RecLock() e o SoftLock() do registro.  
    Endif  
Endif
```



*Anotações*

---

---

---

---

---

## **USED()**

---

A função USED() é utilizada para determinar se há um arquivo de banco de dados em uso em uma área de trabalho específica. O padrão é que USED() opere na área de trabalho correntemente selecionada.

**Sintaxe:** USED()

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Lógico</b>	Verdadeiro (.T.) caso haja um arquivo de banco de dados em uso; caso contrário, retorna falso (.F.).
---------------	--

**Exemplo:**

```
USE Customer NEW  
CONOUT(USED()) // Resulta: .T.  
CLOSE  
CONOUT(USED()) // Resulta: .F.
```

## **ZAP**

---

O comando ZAP remove fisicamente todos os registro da tabela corrente. É necessário que o alias em questão seja aberto em modo exclusivo para esta operação ser realizada.

**Sintaxe:** ZAP

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
cTabela := RetSqlName("SA4")  
cAlias := "TMP"  
USE (cTabela) ALIAS (cAlias) EXCLUSIVE NEW VIA "TOPCONN"  
IfNetErr()  
    MsgStop("Nao foi possivel abrir "+cTabela+" em modo EXCLUSIVO.")  
Else  
    ZAP  
    USE  
    MsgStop("Registros da tabela "+cTabela+" eliminados com sucesso.")  
Endif
```

## Controle de numeração seqüencial

### GETSXENUM()

Obtém o número seqüência do alias especificado no parâmetro, através da referência aos arquivos de sistema SXE/SXF ou ao servidor de numeração, quando esta configuração está habilitada no ambiente Protheus.

- Sintaxe:** GETSXENUM(**cAlias, cCampo, cAliasSXE, nOrdem**)
- Parâmetros**

<b>cAlias</b>	Alias de referência da tabela para a qual será efetuado o controle da numeração seqüencial.
<b>cCampo</b>	Nome do campo no qual está implementado o controle da numeração.
<b>cAliasSXE</b>	Parâmetro opcional, quando o nome do alias nos arquivos de controle de numeração não é o nome convencional do alias para o sistema ERP.
<b>nOrdem</b>	Número do índice para verificar qual a próxima ocorrência do número.

### CONFIRMSXE()

Confirma o número alocado através do último comando GETSXENUM().

- Sintaxe:** CONFIRMSXE(**IVerifica**)
- Parâmetros**

<b>IVerifica</b>	Verifica se o número confirmado não foi alterado, e por consequência já existe na base de dados.
------------------	--

### ROLLBACKSXE()

Descarta o número fornecido pelo último comando GETSXENUM(), retornando a numeração disponível para outras conexões.

- Sintaxe:** ROLLBACKSXE()
- Parâmetros**

<b>Nenhum</b>	.
---------------	---



Anotações

---

---

---

---

## **Validação**

### **ALLWAYSFALSE()**

A função AllwaysFalse() foi criada com o objetivo de compatibilidade, sendo que sempre irá retornar um valor lógico falso, facilitando a especificação desta situação nas parametrizações de validações de modelos de interface pré-definidos no sistema.

- Sintaxe: ALLWAYSFALSE()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Lógico</b>	Retorna um valor lógico sempre falso.
---------------	---------------------------------------

### **ALLWAYSTRUE()**

A função AllwaysTrue() foi criada com o objetivo de compatibilidade, sendo que sempre irá retornar um valor lógico verdadeiro, facilitando a especificação desta situação nas parametrizações de validações de modelos de interface pré-definidos no sistema.

- Sintaxe: ALLWAYSTRUE()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Lógico</b>	Retorna um valor lógico sempre verdadeiro.
---------------	--

### **EXISTCHAV()**

Retorna .T. ou .F. se o conteúdo especificado existe no alias especificado. Caso exista será exibido um help de sistema com um aviso informando da ocorrência.

Função utilizada normalmente para verificar se um determinado código de cadastro já existe na tabela na qual a informação será inserida, como por exemplo o CNPJ no cadastro de clientes ou fornecedores.

- Sintaxe: ExistChav(cAlias, cConteudo, nIndice)**

- Parâmetros**

<b>cAlias</b>	Alias de referência para a validação da informação.
<b>cConteudo</b>	Chave a ser pesquisada, sem a filial.
<b>nIndice</b>	Índice de busca para consulta da chave.

## **EXISTCPO()**

---

Retorna .T. ou .F. se o conteúdo especificado não existe no alias especificado. Caso não exista será exibido um help de sistema com um aviso informando da ocorrência.

Função utilizada normalmente para verificar se a informação digitada em um campo, a qual depende de outra tabela, realmente existe nesta outra tabela, como por exemplo o código de um cliente em um pedido de venda.

- Sintaxe:** ExistCpo(cAlias, cConteudo, nIndice)
- Parâmetros**

<b>cAlias</b>	Alias de referência para a validação da informação.
<b>cConteudo</b>	Chave a ser pesquisada, sem a filial.
<b>nIndice</b>	Índice de busca para consulta da chave.

## **LETTERORNUM()**

---

A função LETTERORNUM() avalia se um determinado conteúdo é composto apenas de letras e números (alfanumérico).

- Sintaxe:** LETTERORNUM(cString)

- Parâmetros:**

<b>cString</b>	String que terá seu conteúdo avaliado.
----------------	--

- Retorno:**

<b>Lógico</b>	Indica que se a string avaliada contém apenas letras e número, ou seja, alfanumérico.
---------------	---

## **NAOVAZIO()**

---

Retorna .T. ou .F. se o conteúdo do campo posicionado no momento não está vazio.

- Sintaxe:** NaoVazio()
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

## **NEGATIVO()**

---

Retorna .T. ou .F. se o conteúdo digitado para o campo é negativo.

- Sintaxe:** Negativo()
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

## **PERTENCE()**

---

Retorna .T. ou .F. se o conteúdo digitado para o campo está contido na string definida como parâmetro da função. Normalmente utilizada em campos com a opção de combo, pois caso contrário seria utilizada a função ExistCpo().

- Sintaxe: Pertence(cString)**
- Parâmetros**

<b>cString</b>	String contendo as informações válidas que podem ser digitadas para um campo.
----------------	---

## **POSITIVO()**

---

Retorna .T. ou .F. se o conteúdo digitado para o campo é positivo.

- Sintaxe: Positivo()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

## **TEXTO()**

---

Retorna .T. ou .F. se o conteúdo digitado para o campo contém apenas números ou alfanuméricos.

- Sintaxe: Texto()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

## **VAZIO()**

---

Retorna .T. ou .F. se o conteúdo do campo posicionado no momento está vazio.

- Sintaxe: Vazio()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---



*Anotações*

---

---

---

---

---

## Manipulação de parâmetros do sistema

### GETMV()

Retorna o conteúdo do parâmetro especificado no arquivo SX6, considerando a filial parametrizada na conexão. Caso o parâmetro não exista será exibido um help do sistema informando a ocorrência.

- Sintaxe:** GETMV(**cParametro**)
- Parâmetros**

<b>cParametro</b>	Nome do parâmetro do sistema no SX6, sem a especificação da filial de sistema.
-------------------	--

### GETNEWPAR()

Retorna o conteúdo do parâmetro especificado no arquivo SX6, considerando a filial parametrizada na conexão. Caso o parâmetro não exista será exibido um help do sistema informando a ocorrência.

Difere do SuperGetMV() pois considera que o parâmetro pode não existir na versão atual do sistema, e por consequência não será exibida a mensagem de help.

- Sintaxe:** GETNEWPAR(**cParametro, cPadrao, cFilial**)
- Parâmetros**

<b>cParametro</b>	Nome do parâmetro do sistema no SX6, sem a especificação da filial de sistema.
<b>cPadrao</b>	Conteúdo padrão que será utilizado caso o parâmetro não exista no SX6.
<b>cFilial</b>	Define para qual filial será efetuada a consulta do parâmetro. Padrão → filial corrente da conexão.



Anotações

---

---

---

---

## **PUTMV()**

---

Atualiza o conteúdo do parâmetro especificado no arquivo SX6, de acordo com as parametrizações informadas.

- Sintaxe: PUTMV(cParametro, cConteudo)**
- Parâmetros**

<b>cParametro</b>	Nome do parâmetro do sistema no SX6, sem a especificação da filial de sistema.
<b>cConteudo</b>	Conteúdo que será atribuído ao parâmetro no SX6.

## **SUPERGETMV()**

---

Retorna o conteúdo do parâmetro especificado no arquivo SX6, considerando a filial parametrizada na conexão. Caso o parâmetro não exista será exibido um help do sistema informando a ocorrência.

Difere do GetMv() pois os parâmetros consultados são adicionados em uma área de memória, que permite que em uma nova consulta não seja necessário acessar e pesquisar o parâmetro na base de dados.

- Sintaxe: SUPERGETMV(cParametro , lHelp , cPadrao , cFilial)**
- Parâmetros**

<b>cParametro</b>	Nome do parâmetro do sistema no SX6, sem a especificação da filial de sistema.
<b>lHelp</b>	Se será exibida a mensagem de Help caso o parâmetro não seja encontrado no SX6.
<b>cPadrao</b>	Conteúdo padrão que será utilizado caso o parâmetro não exista no SX6.
<b>cFilial</b>	Define para qual filial será efetuada a consulta do parâmetro. Padrão → filial corrente da conexão.



*Anotações*

---

---

---

---

---

## Controle de impressão

### AVALIMP()

A função AVALIMP() é utilizada em relatórios específicos em substituição da função CABEC(), configurando a impressora de acordo com o driver escolhido e os parâmetros de impressão disponíveis no array aReturn, respeitando o formato utilizado pela função SETPRINT().

**Sintaxe: AVALIMP(nLimite)**

**Parâmetros:**

<b>nLimite</b>	Tamanho do relatório em colunas, podendo assumir os valores 80,132 ou 220 colunas, respectivamente para os formatos "P", "M" ou "G" de impressão.
----------------	---

**Retorno:**

<b>Caracter</b>	String com caracteres de controle, dependente das configurações escolhidas pelo usuário e do arquivo de driver especificado.
-----------------	--

**Exemplo:**

```
/*
+-----+
| Função      | XAVALIMP      | Autor | ARNALDO RAYMUNDO JR. | Data | 01.01.2007 |
+-----+
| Descrição   | Exemplo de utilização da função AXCADASTRO()           |
|-----+
| Uso         | Curso ADVPL               |
|-----+
*/
USER FUNCTION XAVALIMP()

LOCAL cTitulo      := PADC("AVALIMP", 74)
LOCAL cDesc1       := PADC("Demonstração do uso da função AVALIMP()", 74)
LOCAL cDesc2       := ""
LOCAL cDesc3       := PADC("CURSO DE ADVPL", 74)
LOCAL cTamanho     := "G"
LOCAL cLimite      := 220
LOCAL cNatureza    := ""
LOCAL aReturn      := {"Especial", 1,"Administração", 1, 2, 2,"",1}
LOCAL cNomeProg    := "RAVALIMP"
LOCAL cPerg        := PADR("RAVALIMP",10) // Compatibilização com MP10
LOCAL nLastKey     := 0
LOCAL cString       := "SF2"

Pergunte(cPerg,.F.) // Pergunta no SX1
wnrel:= SetPrint(cString,wnrel,cPerg,cTitulo,cDesc1,cDesc2,cDesc3,.T.)
SetDefault(aReturn,cString)
```

**Exemplo (continuação):**

```
If nLastKey == 27
    Return
Endif

RptStatus({||| RunReport(cString)},cTitulo)
Return

/*
+-----+
| Função | RUNREPORT | Autor | ----- | Data | 01.01.2007 |
+-----+
| Descrição | Função interna de processamento utilizada pela XAVALIMP() |
|+-----+
| Uso | Curso ADVPL
|+-----+
*/
Static Function RunReport(cString)
SetPrc(0,0)

//+-----+
//| Chamada da função AVALIMP()
//+-----+
@ 00,00 PSAY AvalImp(220)
dbSelectArea(cString)
dbSeek(xFilial()+mv_par01+mv_par03,.T.)
...
Return
```

## **CABEC()**

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

**Sintaxe:** Cabec(**cTitulo**, **cCabec1**, **cCabec2**, **cNomeProg**, **nTamanho**, **nCompress**, **aCustomText**, **IPerg**, **cLogo**)

**Parâmetros:**

<b>cTitulo</b>	Título do relatório
<b>cCabec1</b>	String contendo as informações da primeira linha do cabeçalho
<b>cCabec2</b>	String contendo as informações da segunda linha do cabeçalho
<b>cNomeProg</b>	Nome do programa de impressão do relatório.
<b>nTamanho</b>	Tamanho do relatório em colunas (80, 132 ou 220)
<b>nCompress</b>	Indica se impressão será comprimida (15) ou normal (18).
<b>aCustomText</b>	Texto específico para o cabeçalho, substituindo a estrutura padrão do sistema.
<b>IPerg</b>	Permite a supressão da impressão das perguntas do relatório, mesmo que o parâmetro MV_IMPSX1 esteja definido como "S"

**Parâmetros (continuação):**

<b>cLogo</b>	Redefine o bitmap que será impresso no relatório, não necessitando que ele esteja no formato padrão da Microsiga: "LGRL"+SM0->M0_CODIGO+SM0->M0_CODFIL+".BMP"
--------------	--

**Retorno:**

<b>Nenhum</b>	.
---------------	---

**Exemplo:**

```
#INCLUDE "protheus.ch"

/*
+-----+
| Função      | MPTR001          | Autor | ARNALDO RAYMUNDO JR. | Data | 01.01.2007 |
+-----+
| Descrição   | Exemplo de utilização das funções de impressão CABEC() |
|+-----+
| Uso         | Curso ADVPL          |
|+-----+
*/
User Function MPTR001()

Local cDesc1      := "Este programa tem como objetivo imprimir relatorio "
Local cDesc2      := "de acordo com os parametros informados pelo usuario."
Local cDesc3      := "Listagem de clientes"
Local cTitulo      := "Listagem de clientes"
Local lImprime     := .T.

// Parametros da SetPrint()
Local cString      := "SA1"
Local cPerg        := ""
Local lDic         := .T. // Habilita a visualizacao do dicionario
Local aOrd         := RetSixOrd(cString)
Local lCompres     := .T. // .F. - Normal / .T. - Comprimido
Local lFilter       := .T. // Habilita o filtro para o usuario
Local cNomeProg    := "MPTR002"
Local cTamanho      := "M"
Local nTipo         := 18
Local nLimite       := 132

Default lCriaTrab := .T.

Private lEnd        := .F.
Private lAbortPrint := .F.
Private aReturn      := { "Zebrado", 1, "Administracao", 2, 2, 1, "", 1}
//aReturn[4] 1- Retrato, 2- Paisagem
//aReturn[5] 1- Em Disco, 2- Via Spool, 3- Direto na Porta, 4- Email

Private nLastKey   := 0
Private m_pag        := 01
Private wnrel        := "MPTR002"

dbSelectArea("SA1")
dbSetOrder(1)
```

**Exemplo (continuação) :**

**Exemplo (continuação) :**

```
While !EOF()  
  
    If lAbortPrint .OR. nLastKey == 27  
        @nLin,00 PSAY "**** CANCELADO PELO OPERADOR ****"  
        Exit  
    Endif  
  
    If nLin > 55 // Salto de Página. Neste caso o formulario tem 55 linhas...  
        Cabec(cTitulo,cCabec1,cCabec2,cNomeProg,cTamanho,nTipo)  
        nLin := 9  
    Endif  
    ...
```

### **IMPCADAST()**

A função IMPCADAST() cria uma interface simples que permite a impressão dos cadastros do sistema com parametrização DE/ATE.

- Sintaxe:** **IMPCADAST(cCab1, cCab2, cCab3, cNomeProg, cTam, nLimite, cAlias)**

- Parâmetros:**

<b>cCab1</b>	Primeira linha do cabeçalho
<b>cCab2</b>	Segunda linha do cabeçalho
<b>cCab3</b>	Terceira linha do cabeçalho
<b>cNomeProg</b>	Nome do programa
<b>cTam</b>	Tamanho do relatório nos formatos "P", "M" e "G".
<b>nLimite</b>	Número de colunas do relatório, seguindo o formato especificado no tamanho, aonde: "P"- 80 colunas "M"- 132 colunas "G"- 220 colunas
<b>cAlias</b>	Alias do arquivo de cadastro que será impresso

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **MS\_FLUSH()**

A função MS\_FLUSH() envia o spool de impressão para o dispositivo previamente especificado com a utilização das funções AVALIMP() ou SETPRINT().

- Sintaxe:** **MS\_FLUSH()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

Nenhum

.

**Exemplo:**

```
/*
+-----+
| Função      | RUNREPORT    | Autor | ----- | Data | 01.01.2007 |
+-----+
| Descrição | Função interna de processamento utilizada pela MPTR001() |
|-----+
| Uso        | Curso ADVPL |
|-----+
| Observação| Continuação do exemplo da função CABEC() |
|-----+
*/
Static Function RunReport(cTitulo,cString,cNomeProg,cTamanho,nTipo,nLimite)

Local nLin      := 80
Local cCabec1   := ""
Local cCabec2   := ""
Local cArqInd

cCabec1 := "CODIGO"+Space(2)+"LOJA"+Space(2)+"NOME REDUZIDO"+Space(9)
cCabec1 += "RAZAO SOCIAL"+Space(30)+"CNPJ"+Space(18)+"INSCR.ESTADUAL"+Space(8)
cCabec1 += "CEP"
cCabec2 := "ESTADO"+Space(2)+"MUNICIPIO"+Space(8)+"ENDERECO"

dbSelectArea("TRBSA1")
dbGoTop()

SetRegua(RecCount())

While !EOF()

If lAbortPrint .OR. nLastKey == 27
  @nLin,00 PSAY "**** CANCELADO PELO OPERADOR ****"
  Exit
Endif

If nLin > 55 // Salto de Página. Neste caso o formulario tem 55 linhas...
  Cabec(cTitulo,cCabec1,cCabec2,cNomeProg,cTamanho,nTipo)
  nLin := 9
Endif

// Primeira linha de detalhe:
@nLin,000 PSAY TRBSA1->A1_COD
@nLin,008 PSAY TRBSA1->A1_LOJA
@nLin,014 PSAY TRBSA1->A1_NREDUZ
@nLin,036 PSAY TRBSA1->A1_NOME
@nLin,078 PSAY TRBSA1->A1_CGC
@nLin,100 PSAY TRBSA1->A1_INSCR
@nLin,122 PSAY TRBSA1->A1_CEP
nLin++
```

**Exemplo (continuação):**

```
// Segunda linha de detalhe
@nLin,000 PSAY TRBSA1->A1_EST
@nLin,008 PSAY TRBSA1->A1_MUN
@nLin,025 PSAY TRBSA1->A1_END
nLin++

//Linha separadora de detalhes
@nLin,000 PSAY Replicate("-",nLimite)
nLin++

dbSkip() // Avanca o ponteiro do registro no arquivo
EndDo

SET DEVICE TO SCREEN

If aReturn[5]==1
  dbCommitAll()
  SET PRINTER TO
  OurSpool(wnrel)
Endif

MS_FLUSH()
RETURN
```

## **OURSPOOL()**

A função OURSPOOL() executa o gerenciador de impressão da aplicação Protheus, permitindo a visualização do arquivo de impressão gerado pelo relatório no formato PostScrip® com extensão "##R".

**Sintaxe: OURSPOOL(cArquivo)**

**Parâmetros:**

<b>cArquivo</b>	Nome do relatório a ser visualizado.
-----------------	--------------------------------------

**Retorno:**

.	.
---	---

**Exemplo:**

```
If aReturn[5]==1 // Indica impressão em disco.
  dbCommitAll()
  SET PRINTER TO
  OurSpool(wnrel)
Endif
```

## **RODA()**

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



Dica

Pode ser utilizado o ponto de entrada "RodaEsp" para tratamento de uma impressão específica.

**Sintaxe: Roda(uPar01, uPar02, cSize)**

**Parâmetros:**

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

**Retorno:**

**Nenhum**

.

**Exemplo:**

```
/*
+-----+
| Função | TESTIMPR | Autor | MICROSIGA           | Data | 01.01.2007 |
+-----+
| Descrição | Exemplo de utilização da função RODA() em conjunto com a CABEC. |
+-----+
| Uso       | Curso ADVPL                         |
+-----+
*/
#include "protheus.ch"

User Function TestImpr()
Local wnrel
Local cString := "SA1"
Local titulo := "Teste Impressão de Relatorios"
Local NomeProg := "XXX"
Local Tamanho := "M"

PRIVATE aReturn := { "Zebrado", 1,"Administracao", 1, 2, 1, "",1 }

wnrel:=SetPrint(cString,NomeProg,"",@titulo,"", "", "", .F.,.F.,.F.,Tamanho,,.F.)
SetDefault(aReturn,cString)

RptStatus({|lEnd| TestRel(@lEnd,wnRel,cString,Tamanho,NomeProg)},titulo)

Return
```

**Exemplo (continuação):**

```
/*
+-----+
| Função | TESTREL      | Autor | MICROSIGA           | Data | 01.01.2007 |
+-----+
| Descrição | Função interna de impressão da TestImpr(). |
|-----|
| Uso       | Curso ADVPL          |
|-----|
*/
User Function TestRel(lEnd,WnRel,cString,Tamanho,NomeProg)

LOCAL cabec1,cabec2
LOCAL cRodaTxt := oemtoansi("Rodapé")
Local nCntImpr
Local nTipo

nCntImpr := 0
li := 80
m_pag := 1
nTipo := 15
titulo:= oemtoansi("Lista de Clientes")
cabec1:= oemtoansi("COD LOJA NOME"+Space(27)+ "NOME FANTASIA")
cabec2:=""

dbSelectArea("SA1")
dbGoTop()
SetRegua(LastRec())
While !Eof()
    IncRegua()
    If Li > 60
        cabec(titulo,cabec1,cabec2,nomeprog,tamanho,15)
        @ Li,0 PSAY __PrtThinLine()
    Endif
    nCntImpr++
    Li++
    @ Li,01 PSAY A1_COD
    @ Li,05 PSAY A1_LOJA
    @ Li,10 PSAY A1_NOME
    @ Li,51 PSAY A1_NREDUZ
    If Li > 60
        Li:=66
    Endif
    dbSkip()
EndDO

If li != 80
    Roda(nCntImpr,cRodaTxt,Tamanho)
EndIf

Set Device to Screen
If aReturn[5] = 1
    Set Printer To
    dbCommitAll()
    OurSpool(wnrel)
Endif
MS_FLUSH()
Return
```

## **SETDEFAULT()**

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

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

**Parâmetros:**

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

**Retorno:**

<b>Nenhum</b>	.
---------------	---

**Estrutura aReturn:**

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



*Anotações*

---

---

---

---

## **SETPRC()**

---

A função SETPRC() é utilizada para posicionar o dispositivo de impressão ativo, previamente definido pelo uso das funções AVALIMP() ou SETPRINT() , em uma linha/coluna especificada.

- Sintaxe:** **SETPRC(nLinha, nColuna)**

- Parâmetros:**

<b>nLinha</b>	Linha na qual deverá ser posicionado o dispositivo de impressão.
<b>nColuna</b>	Coluna na qual deverá ser posicionado o dispositivo de impressão.

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
aReturn := { "", 1, "", 2, 3, cPorta , "",IndexOrd() }
SetPrint(Alias(),"","","","",F.....'EPSON.DRV',.T.,,cPorta)
if nLastKey == 27
Return (.F.)
Endif
SetDefault(aReturn,Alias())
SetPrc(0,0)
```

## **SETPRINT()**

---

A função SetPrint() cria uma interface padrão onde as opções de impressão de um relatório podem ser configuradas. Basicamente duas variáveis m\_pag e aReturn precisam ser declaradas como privadas (private) antes de executar a SetPrint(), sendo que:

- m\_pag**: controla o número de páginas.
- aReturn**: vetor contendo as opções de impressão, sendo sua estrutura básica composta de 8 (oito) elementos.

Após confirmada, os dados são armazenados no vetor aReturn que será passado como parâmetro para função SetDefault().

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

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo a ser impresso.
<b>cProgram</b>	Nome do arquivo a ser gerado em disco.
<b>cPergunte</b>	Grupo de perguntas cadastrado no dicionário SX1.
<b>cTitle</b>	Título do relatório.

**Parâmetros (continuação):**

<b>cDesc1</b>	Descrição do relatório.
<b>cDesc2</b>	Continuação da descrição do relatório.
<b>cDesc3</b>	Continuação da descrição do relatório.
<b>IDic</b>	Utilizado na impressão de cadastro genérico permite a escolha dos campos a serem impressos. Se o parametro cAlias não for informado o valor padrão assumido será .F.
<b>aOrd</b>	Ordem(s) de impressão.
<b>ICompres</b>	Se verdadeiro (.T.) permite escolher o formato da impressão, o valor padrão assumido será .T.
<b>cSize</b>	Tamanho do relatório "P", "M" ou "G".
<b>uParm12</b>	Parâmetro reservado
<b>IFilter</b>	Se verdadeiro (.T.) permite a utilização do assistente de filtro, o valor padrão assumido será .T.
<b>ICrystal</b>	Se verdadeiro (.T.) permite integração com Crystal Report, o valor padrão assumido será .F.
<b>cNameDrv</b>	Nome de um driver de impressão.
<b>uParm16</b>	Parâmetro reservado.
<b>IServer</b>	Se verdadeiro (.T.) força impressão no servidor.
<b>cPortPrint</b>	Define uma porta de impressão padrão.

**Retorno:**

<b>Caracter</b>	Nome do Relatório
-----------------	-------------------

**Estrutura aReturn:**

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

## Controle de processamentos

### **ABREEXCL()**

A função ABREEXCL() fecha o arquivo cujo alias está expresso em <cAlias> e o reabre em modo exclusivo para proceder operações em que isto é necessário, como por exemplo, PACK. Se outra estação estiver usando o arquivo, o retorno será .F..

- Sintaxe:** ABREEXCL(cAlias)

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo que será re-aberto em modo exclusivo.
---------------	--

- Retorno:**

<b>Lógico</b>	Indica se foi possível abrir o arquivo em modo exclusivo.
---------------	---

### **CLOSEOPEN()**

A função CLOSEOPEN() é utilizada para fechar e re-abrir uma lista de arquivos especificada.

- Sintaxe:** CLOSEOPEN(aClose, aOpen)

- Parâmetros:**

<b>aClose</b>	Array contendo os Aliases dos arquivos que deverão ser fechados.
<b>aOpen</b>	Array contendo os Aliases dos arquivos que deverão ser abertos.

- Retorno:**

<b>Lógico</b>	Indica se todos os arquivos especificados em aOpen foram abertos com sucesso.
---------------	---

### **CLOSESFILE()**

A função CLOSESFILE() fecha todos os arquivos em uso pela conexão, com exceção dos SXs (inclusive SIX), SM2 e SM4.

- Sintaxe:** CLOSESFILE(cAlias)

- Parâmetros:**

<b>cAlias</b>	String contendo os nomes dos demais Aliases que não deverão ser fechados, separando os itens com "/".
---------------	---

- Retorno:**

<b>Lógico</b>	Indica se todos os arquivos foram fechados com sucesso.
---------------	---

## **CHKFILE()**

A função CHKFILE() retorna verdadeiro (.T.) se o arquivo já estiver aberto ou se o Alias não for informado. Sempre que desejar mudar o modo de acesso do arquivo (de compartilhado para exclusivo ou vice-versa), feche-o antes de chamá-la.

- Sintaxe:** ChkFile(cAlias,!Excl,cNewAlias)

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo a ser re-aberto.
<b>!Excl</b>	Se for informado verdadeiro (.T.), o arquivo será aberto em modo exclusivo, caso contrário, o arquivo será aberto em modo compartilhado. Se este parâmetro não for informado, será assumido falso (.F.).
<b>cNewAlias</b>	Novo Alias para re-abertura do arquivo.

- Retorno:**

<b>Lógico</b>	Indica se o arquivo foi re-aberto com sucesso.
---------------	--

### **Exemplo:**

```
dbSelectArea("SA1")
dbCloseArea()
IOk := .T.
While .T.
    IF !ChkFile("SA1",.T.)
        nResp := Alert("Outro usuário usando! Tenta de novo?", {"Sim", "Nao"})
        If nResp == 2
            IOk := .F.
            Exit
        Endif
    Endif
EndDo
If IOk
    // Faz o processamento com o arquivo...
Endif
// Finaliza
If Select("SA1")
    dbCloseArea()
Endif
ChkFile("SA1",.F.)
Return
```



*Anotações*

---

---

---

---

## **CONOUT()**

---

A função CONOUT() permite a exibição de uma mensagem de texto no console do Server do Protheus. Caso o Protheus esteja configurado como serviço a mensagem será gravada no arquivo de log.

- Sintaxe:** **CONOUT(cMensagem)**

- Parâmetros:**

<b>cMensagem</b>	String contendo a mensagem que deverá ser exibida no console do Protheus.
------------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **CRIAVAR()**

---

A função CRIAVAR() cria uma variável, retornando o valor do campo, de acordo com o dicionário de dados, inclusive avaliando o inicializador padrão, permitindo um retorno de acordo com o tipo de dado definido no dicionário.

- Sintaxe:** **CriaVar(cCampo,lIniPad,cLado)**

- Parâmetros:**

<b>cCampo</b>	Nome do campo
<b>lIniPad</b>	Indica se considera (.T.) ou não (.F.) o inicializador
<b>cLado</b>	Se a variável for caracter, cLado pode ser: "C" - centralizado, "L" - esquerdo ou "R" – direito.

- Retorno:**

<b>Indefinido</b>	Tipo de dado de acordo com o dicionário de dados, considerando inicializador padrão
-------------------	---

### **Exemplo:**

```
// Exemplo do uso da função CriaVar:  
cNumNota := CriaVar("F2_DOC") // Retorna o conteúdo do  
// inicializador padrão,  
// se existir, ou espaços em branco  
Alert(cNumNota)  
Return
```

## **DISARMTRANSACTION()**

A função DISARMTRANSACTION() é utilizada para realizar um “RollBack” de uma transação aberta com o comando BEGIN TRANSACTION e delimitada com o comando END TRANSACTION.

Ao utilizar esta função, todas as alterações realizadas no intervalo delimitado pela transação são desfeitas, restaurando a situação da base de dados ao ponto imediatamente anterior ao início do processamento.



**Importante**

O uso da função DISARMTRANSACTION() não finaliza a conexão ou o processamento corrente.

Caso seja necessário além de desfazer as alterações, finalizar o processamento, deverá ser utilizada a função USEREXCEPTION().

- Sintaxe: DISARMTRANSACTION()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
IMsErroAuto := .F.  
MSExecAuto({|x,y| MATA240(x,y)}, aCampos, 3)  
  
If IMsErroAuto  
  
    aAutoErro := GETAUTOGRLLOG()  
    DisarmTransaction()  
    MostraErro()  
  
EndIf
```



**Anotações**

---

---

---

---

## **EXECBLOCK()**

A função EXECBLOCK() executa uma função de usuário que esteja compilada no repositório. Esta função é normalmente utilizada pelas rotinas padrões da aplicação Protheus para executar pontos de entrada durante seu processamento.



*Importante*

A função de usuário executada através da EXECBLOCK() não recebe parâmetros diretamente, sendo que estes estarão disponíveis em uma variável private denominada **PARAMIXB**.



*Importante*

A variável **PARAMIXB** é o reflexo do parâmetro xParam, definido na execução da função EXECBLOCK(). Caso seja necessária a passagem de várias informações, as mesmas deverão ser definidas na forma de um array, tornando **PARAMIXB** um array também, a ser tratado na função de usuário que será executada.

## **EXISTBLOCK()**

A função EXISTBLOCK() verifica a existência de uma função de usuário compilada no repositório de objetos da aplicação Protheus. Esta função é normalmente utilizada nas rotinas padrões da aplicação Protheus para determinar a existência de um ponto de entrada e permitir sua posterior execução.

- Sintaxe:** **EXISTBLOCK(cFunção)**

- Parâmetros:**

<b>cFunção</b>	Nome da função que será avaliada.
----------------	-----------------------------------

- Retorno:**

<b>Lógico</b>	Indica se a função de usuário existe compilada no repositório de objetos corrente.
---------------	--

### **Exemplo:**

```
IF EXISTBLOCK("MT100GRV")
    EXECBLOCK("MT100GRV",.F.,.F.,aParam)
ENDIF
```

- Sintaxe: EXECBLOCK(cFunção, IReserv1, IReserv2, xParam)**

- Parâmetros:**

<b>cFunção</b>	Nome da função de usuário que será executada.
<b>IReserv1</b>	Parâmetro de uso reservado da aplicação. Definir como .F.
<b>IReserv2</b>	Parâmetro de uso reservado da aplicação. Definir como .F.
<b>xParam</b>	Conteúdo que ficará disponível na função de usuário executada, na forma da variável private <b>PARAMIXB</b> .

- Retorno:**

<b>Indefinido</b>	O retorno da EXECBLOCK() é definido pela função que será executada.
-------------------	---

### **Exemplo:**

```
aParam := {cNota, cSerie, cFornece, cLoja}

IF EXISTBLOCK("MT100GRV")
  IGravou := EXECBLOCK("MT100GRV",.F.,.F.,aParam)
ENDIF

USER FUNCTION MT100GRV()

LOCAL cNota := PARAMIXB[1]
LOCAL cSerie:= PARAMIXB[1]
LOCAL cFornece:= PARAMIXB[1]
LOCAL cLoja:= PARAMIXB[1]

RETURN .T.
```

### **ERRORBLOCK()**

A função ERRORBLOCK() efetua o tratamento de erros e define a atuação de um handler de erros sempre que ocorrer um erro em tempo de execução. O manipulador de erros é especificado como um bloco de código da seguinte forma:

- { |<objError>| <lista de expressões>, ... }, onde:

<objError> é um error object que contém informações sobre o erro. Dentro do bloco de código, podem ser enviadas mensagens ao error object para obter informações sobre o erro. Se o bloco de tratamento de erros retornar verdadeiro (.T.), a operação que falhou é repetida, e se retornar falso (.F.), o processamento recomeça.

Se não foi especificado nenhum <bErrorHandler> utilizando ERRORBLOCK() e ocorrer um erro em tempo de execução, o bloco de tratamento ao de erros padrão é avaliado. Este manipulador de erros exibe uma mensagem descritiva na tela, ajusta a função ERRORLEVEL() para 1, e depois sai do programa (QUIT).

Como ERRORBLOCK() retorna o bloco de tratamento ao de erros corrente, é possível especificar um bloco de tratamento de erros para uma operação gravando-se o bloco de manipulação de erros corrente e depois recuperando-o após o final da operação. Além disso,

uma importante consequência do fato de os blocos de tratamento de erros serem especificados como blocos de código, é que eles podem ser passados pararotinas e funções definidas por usuário e depois retornadas como valores.

- Sintaxe: ERRORBLOCK ( < bErrorHandler > )**
  - Parâmetros:**

<b>bErrorHandler</b>	O bloco de código a ser executado toda vez que ocorrer um erro em tempo de execução. Quando avaliado, o <code>&lt;bErrorHandler&gt;</code> é passado na forma de um objeto erro como um argumento pelo sistema.
----------------------	--

- Retorno:**

<b>Code-block</b>	Retorna o bloco de código corrente que tratar o erro. Caso não tenha sido enviado nenhum bloco de tratamento de erro desde que o programa foi invocado, ERRORBLOCK() retorna o bloco de tratamento de erro padrão.
-------------------	--

### **Exemplo:**

```

Function CA010Form()
LOCAL xResult
LOCAL cForm:= Upper(&(ReadVar()))
LOCAL bBlock:= ErrorBlock( { |e| ChecErro(e) } )
LOCAL cOutMod
Local IOptimize := GetNewPar("MV_OPTNFE",.F.) .Or. GetNewPar("MV_OPTNFS",.F.)

PRIVATE IRet:=.T.

cVarOutMod := If(Type("cVarOutMod") = "U", "", cVarOutMod)
cOutMod     := cVarOutMod + If(Right(cVarOutMod, 1) = ",", "", ",") 

While ! Empty(cOutMod)
    If Left(cOutMod, At(",", cOutMod) - 1) $ Upper(cForm)           // no modulo
        Help( " ",1,"ERR_FORM,"Variavel nao disponivel para este modulo"
        Return .F.
    Endif
    cOutMod := Subs(cOutMod, At(",", cOutMod) + 1)
EndDo
If ("LERSTR"$cForm .or. "LERVAL"$cForm .or. "LERDATA"$cForm) .And. M->I5_CODIGO >
"499"
    Help( " ",1,"CA010TXT")
    ErrorBlock(bBlock)
    Return .F.
Endif
BEGIN SEQUENCE
    If !"EXECBLOCK"$cForm .and. !"LERSTR"$cForm .And.; // nao executa execblock
        !"LERVAL"$cForm .And.; // nem funcao de leitura
        !"LERDATA"$cForm .And.; // de texto no cadastramento
        IIf(!IOptimize,.T.,!"CTBANFS"$cForm .And. !"CTBANFE"$cForm)
        xResult := &cForm
    Endif
END SEQUENCE
ErrorBlock(bBlock)
Return IRet

```

## **FINAL()**

A função FINAL() executa as operações básicas que garantem a integridade dos dados ao finalizar o sistema desmontando as transações (se houver), desbloqueando os semáforos e fechando as tabelas abertas, finalizando-o em seguida.

- Sintaxe:** Final( [cMensagem1], [cMensagem2] )

- Parâmetros:**

<b>cMensagem1</b>	Primeira mensagem
<b>cMensagem2</b>	Segunda mensagem

- Retorno:**

<b>Nenhum</b>	.
---------------	---

### **Exemplo:**

```
User Function ValidUser( cUsuario, cSenha )  
  
Local cMensag1 := "Usuário invalido!"  
Local cMensag2 := "Opção disponível para usuários Administradores!"  
  
If !PswAdmin( cUsuario, cSenha )  
    Final( cMensag1, cMensag2 )  
EndIf  
  
Return
```

## **FINDFUNCTION()**

A função FINDFUNCTION() tem como objetivo verificar se uma determinada função se encontra no repositório de objetos e até mesmo do binário do Protheus, sendo uma função básica da linguagem.

- Sintaxe:** FINDFUNCTION(cFunção)

- Parâmetros:**

<b>cFunção</b>	Nome da função que será avaliada no repositório de objetos corrente.
----------------	--

- Retorno:**

<b>Lógico</b>	Indica se a função existe compilada no repositório de objetos corrente.
---------------	---

## **FUNDESC()**

---

A função FunDesc() retornará a descrição de uma opção selecionada no menu da aplicação.

- Sintaxe: FUNDESC()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Caracter</b>	Descrição da opção selecionada no menu da aplicação.
-----------------	--

## **FUNNAME()**

---

A função FunName() retornará o nome de uma função executada a partir de um menu da aplicação.

- Sintaxe: FUNNAME()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Caracter</b>	Nome da função executada a partir do menu da aplicação.
-----------------	---

## **GETAREA()**

---

Função utilizada para proteger o ambiente ativo no momento de algum processamento específico. Para salvar uma outra área de trabalho (alias) que não o ativo, a função GetArea() deve ser executada dentro do alias: ALIAS->(GetArea()).

- Sintaxe: GETAREA()**

- Retorno: Array contendo {Alias(),IndexOrd(),Recno()}**

- Parâmetros**

<b>Nenhum</b>	.
---------------	---

## **GETCOUNTRYLIST()**

A função GETCOUNTRYLIST() retorna um array de duas dimensões contendo informações dos países localizados.

- Sintaxe:** **GetCountryList()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Array</b>	Array de duas dimensões, sendo uma linha para cada país localizado, contendo em cada posição a sigla dos países, o nome do país e a identificação do país com dois dígitos.
--------------	---

### **Exemplo:**

```
Local aArray := GetCountryList()
Local cSigla := GetMv( "MV_PAISLOC" )
Local nPos

nPos := Ascan( aArray, {|d| d[1] == Upper(cSigla) } )
If nPos > 0
    APMsgInfo( "País de localização " + aArray[nPos,2] )
EndIf
```

## **ISINCALLSTACK()**

A função ISINCALLSTACK() verifica se uma determinada função está existindo dentro da pilha de chamadas do processamento corrente.

- Sintaxe:** **IsInCallStack( cIsInCallStack , cStackExit )**

- Parâmetros:**

<b>cIsInCallStack</b>	Nome da função que desejassem pesquisar na pilha.
<b>cStackExit</b>	String que identifica o ponto em que desejassem finalizar a busca. Caso não seja informada, será utilizada como padrão a expressão "STACK_EXIT".

- Retorno:**

<b>Lógico</b>	Indica se a função especificada encontrasse na pilha de chamadas do processamento corrente, até o ponto de saída especificado.
---------------	--

## **REGTOMEMORY()**

Inicializa as variáveis de memória identificadas pelo uso do alias "M->" de acordo com a estrutura e/ou informações contidas no arquivo definido como referência.



*Importante*

A função REGTOMEMORY necessita que o ALIAS exista no SX3 para a operação de inclusão, mesmo que o parâmetro IDic seja definido como falso.

- Sintaxe: REGTOMEMORY(cAlias, lInclui, lDic, lInitPad)**

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo que será utilizado como referência para inicialização das variáveis de memória.
<b>lInclui</b>	Identifica se as variáveis deverão ser inicializadas com conteúdos padrões, ou contendo as informações do registro posicionado do alias especificado.
<b>lDic</b>	Define se irá utilizar o dicionário de dados como base para inicialização das variáveis ou se irá utilizar o comando CRIAVAR().
<b>lInitPad</b>	Define se irá executar o inicializador padrão do campo, contido no dicionário de dados. Parâmetro utilizado pela CRIAVAR().

- Retorno:**

**Nenhum**

.



*Anotações*

---

---

---

---

## **RESTAREA()**

Função utilizada para devolver a situação do ambiente salva através do comando GETAREA(). Deve-se observar que a última área restaurada é a área que ficará ativa para a aplicação.

- Sintaxe: RESTAREA(aArea)**
- Parâmetros**

<b>aArea</b>	Array contendo: {cAlias, nOrdem, nRecno}, normalmente gerado pelo uso da função GetArea().
--------------	--

### **Exemplo:**

```
// ALIAS ATIVO ANTES DA EXECUÇÃO DA ROTINA → SN3
User Function XATF001()

LOCAL cVar
LOCAL aArea := GetArea()
LOCAL IRet := .T.

cVar := &(ReadVar())

dbSelectArea("SX5")
IF !dbSeek(xFilial()+"Z1"+cVar)

    cSTR0001 := "REAV - Tipo de Reavaliacao"
    cSTR0002 := "Informe um tipo de reavaliacao valido"
    cSTR0003 := "Continuar"
    Aviso(cSTR0001,cSTR0002,{cSTR0003},2)
    IRet := .F.

ENDIF

RestArea(aArea)
Return( IRet )
```

## **USEREXCEPTION()**

A função USEREXCEPTION() tem o objetivo de forçar um erro em ADVPL de forma que possamos tratar de alguma forma. USEREXCEPTION() recebe uma string contendo uma descrição do erro, essa descrição será exibida de acordo com o ambiente que se está executando, caso um ambiente ERP, será exibida uma tela de erro.

- Sintaxe: USEREXCEPTION(cMensagem)**

- Parâmetros:**

<b>cMensagem</b>	Mensagem que será exibida no cabeçalho do erro, contendo a explicação da exceção.
------------------	---

- Retorno:**

**Nenhum**



*Anotações*

---

---

---

---

## Utilização de recursos do ambiente ERP

### AJUSTASX1()

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

- Sintaxe: AJUSTASX1(cPerg, aPergs)**

- Parâmetros:**

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

- Retorno:**

<b>Nenhum</b>	.
---------------	---

- Estrutura – Item do array aPerg:**

Posição	Campo	Tipo	Descrição
<b>01</b>	X1_PERGUNT	Caractere	Descrição da pergunta em português
<b>02</b>	X1_PERSPA	Caractere	Descrição da pergunta em espanhol
<b>03</b>	X1_PERENG	Caractere	Descrição da pergunta em inglês
<b>04</b>	X1_VARIAVL	Caractere	Nome da variável de controle auxiliar (mv_ch)
<b>05</b>	X1_TIPO	Caractere	Tipo do parâmetro
<b>06</b>	X1_TAMANHO	Numérico	Tamanho do conteúdo do parâmetro
<b>07</b>	X1_DECIMAL	Numérico	Número de decimais para conteúdos numéricos
<b>08</b>	X1_PRESEL	Numérico	Define qual opção do combo é a padrão para o parâmetro.
<b>09</b>	X1_GSC	Caractere	Define se a pergunta será do tipo G – Get ou C – Choice (combo)
<b>10</b>	X1_VALID	Caractere	Expressão de validação do parâmetro
<b>11</b>	X1_VAR01	Caractere	Nome da variável MV_PAR+“Ordem” do parâmetro
<b>12</b>	X1_DEF01	Caractere	Descrição da opção 1 do combo em português
<b>13</b>	X1_DEFSPA1	Caractere	Descrição da opção 1 do combo em espanhol
<b>14</b>	X1_DEFENG1	Caractere	Descrição da opção 1 do combo em inglês
<b>15</b>	X1_CNT01	Caractere	Conteúdo padrão ou último conteúdo definido como respostas para a pergunta.
<b>16</b>	X1_VAR02	Caractere	Não é informado
<b>17</b>	X1_DEF02	Caractere	Descrição da opção X do combo em português
<b>18</b>	X1_DEFSPA2	Caractere	Descrição da opção X do combo em espanhol
<b>19</b>	X1_DEFENG2	Caractere	Descrição da opção X do combo em inglês
<b>20</b>	X1_CNT02	Caractere	Não é informado
<b>21</b>	X1_VAR03	Caractere	Não é informado
<b>22</b>	X1_DEF03	Caractere	Descrição da opção X do combo em português

**Estrutura – Item do array aPerg (continuação):**

<b>23</b>	X1_DEFSPA3	Caractere	Descrição da opção X do combo em espanhol
<b>24</b>	X1_DEFENG3	Caractere	Descrição da opção X do combo em inglês
<b>25</b>	X1_CNT03	Caractere	Não é informado
<b>26</b>	X1_VAR04	Caractere	Não é informado
<b>27</b>	X1_DEF04	Caractere	Descrição da opção X do combo em português
<b>28</b>	X1_DEFSPA4	Caractere	Descrição da opção X do combo em espanhol
<b>29</b>	X1_DEFENG4	Caractere	Descrição da opção X do combo em inglês
<b>30</b>	X1_CNT04	Caractere	Não é informado
<b>31</b>	X1_VAR05	Caractere	Não é informado
<b>32</b>	X1_DEF05	Caractere	Descrição da opção X do combo em português
<b>33</b>	X1_DEFSPA5	Caractere	Descrição da opção X do combo em espanhol
<b>34</b>	X1_DEFENG5	Caractere	Descrição da opção X do combo em inglês
<b>35</b>	X1_CNT05	Caractere	Não é informado
<b>36</b>	X1_F3	Caractere	Código da consulta F3 vinculada ao parâmetro
<b>37</b>	X1_GRPSXG	Caractere	Código do grupo de campos SXG para atualização automática, quando o grupo for alterado.
<b>38</b>	X1_PYME	Caractere	Se a pergunta estará disponível no ambiente Pyme
<b>39</b>	X1_HELP	Caractere	Conteúdo do campo X1_HELP
<b>40</b>	X1_PICTURE	Caractere	Picture de formatação do conteúdo do campo.
<b>41</b>	aHelpPor	Array	Vetor simples contendo as linhas de help em português para o parâmetro. Trabalhar com linhas de até 40 caracteres.
<b>42</b>	aHelpEng	Array	Vetor simples contendo as linhas de help em inglês para o parâmetro. Trabalhar com linhas de até 40 caracteres.
<b>43</b>	aHelpSpa	Array	Vetor simples contendo as linhas de help em espanhol para o parâmetro. Trabalhar com linhas de até 40 caracteres.



Anotações

---



---



---



---



---

## **ALLUSERS()**

---

A função ALLUSERS() retorna um array multidimensional contendo as informações dos usuários do sistema.

- Sintaxe: ALLUSERS()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Array</b>	Array multidimensional contendo as informações dos usuários do sistema, aonde para cada usuário serão demonstradas as seguintes informações:  aArray[x][1] → Configurações gerais de acesso aArray[x][2] → Configurações de impressão aArray[x][3] → Configurações de acesso aos módulos
--------------	--

- Array de informações dos usuários: Configurações gerais de acesso**

<b>Elemento</b>	<b>Descrição</b>	<b>Tipo</b>	<b>Qtd.</b>
<b>1</b>			
<b>1</b>	ID	C	6
<b>2</b>	Nome	C	15
<b>3</b>	Senha	C	6
<b>4</b>	Nome Completo	C	30
<b>5</b>	Vetor com nº últimas senhas	A	--
<b>6</b>	Data de validade	D	8
<b>7</b>	Quantas vezes para expirar	N	4
<b>8</b>	Autorizado a alterar a senha	L	1
<b>9</b>	Alterar a senha no próximo logon	L	1
<b>10</b>	Vetor com os grupos	A	--
<b>11</b>	ID do superior	C	6
<b>12</b>	Departamento	C	30
<b>13</b>	Cargo	C	30
<b>14</b>	E-Mail	C	130
<b>15</b>	Número de acessos simultâneos	N	4
<b>16</b>	Data da última alteração	D	8
<b>17</b>	Usuário bloqueado	L	1
<b>18</b>	Número de dígitos para o ano	N	1
<b>19</b>	Listner de ligações	L	1
<b>20</b>	Ramal	C	4

**Array de informações dos usuários: Configurações de impressão**

<b>Elemento</b>	<b>Descrição</b>			<b>Tipo</b>	<b>Qtd.</b>
<b>2</b>					
	<b>1</b>	Vetor com horários de acesso	A	--	
	<b>2</b>	Idioma	N	1	
	<b>3</b>	Diretório	C	100	
	<b>4</b>	Impressora	C	--	
	<b>5</b>	Acessos	C	512	
	<b>6</b>	Vetor com empresas	A	--	
	<b>7</b>	Ponto de entrada	C	10	
	<b>8</b>	Tipo de impressão	N	1	
	<b>9</b>	Formato	N	1	
	<b>10</b>	Ambiente	N	1	
	<b>11</b>	Prioridade p/ config. do grupo	L	1	
	<b>12</b>	Opção de impressão	C	50	
	<b>13</b>	Acesso a outros dir de impressão	L	1	

**Array de informações dos usuários: Configurações de acesso aos módulos**

<b>Elemento</b>	<b>Descrição</b>			<b>Tipo</b>	<b>Qtd.</b>
<b>3</b>					
	<b>1</b>	Módulo+nível+menu	C		



*Anotações*

---

---

---

---

---

## **ALLGROUPS()**

A função ALLGROUPS() retorna um array multidimensional contendo as informações dos grupos de usuários do sistema.

**Sintaxe: ALLGROUPS()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Array</b>	Array multidimensional contendo as informações dos grupos de usuários do sistema, aonde para cada grupo serão demonstradas as seguintes informações:  aArray[x][1] → Configurações gerais de acesso aArray[x][2] → Configurações de acesso aos módulos
--------------	---

**Array de informações dos grupos: Configurações gerais de acesso**

<b>Elemento</b>	<b>Descrição</b>		<b>Tipo</b>	<b>Qtd.</b>
<b>1</b>				
<b>1</b>	ID		C	6
<b>2</b>	Nome		C	20
<b>3</b>	Vetor com horários de acesso		A	
<b>4</b>	Data de validade		D	8
<b>5</b>	Quantas vezes para expirar		N	4
<b>6</b>	Autorizado a alterar a senha		L	1
<b>7</b>	Idioma		N	1
<b>8</b>	Diretório		C	100
<b>9</b>	Impressora		C	
<b>10</b>	Acessos		C	512
<b>11</b>	Vetor com empresas		A	
<b>12</b>	Data da última alteração		D	8
<b>13</b>	Tipo de impressão		N	1
<b>14</b>	Formato		N	1
<b>15</b>	Ambiente		N	1
<b>16</b>	Opção de impressão		L	1
<b>17</b>	Acesso a outros Dir de impressão		L	1

**Array de informações dos grupos: Configurações de acesso aos módulos**

<b>Elemento</b>	<b>Descrição</b>		<b>Tipo</b>	<b>Qtd.</b>
2				
	1	Modulo+nível+menu	C	

## **CGC()**

---

A função CGC() valida o CGC digitado, utilizando o algoritmo nacional para verificação do dígito de controle.

- Sintaxe:** CGC(**cCGC**)

- Parâmetros:**

<b>cCGC</b>	String contendo o CGC a ser validado
-------------	--------------------------------------

- Retorno:**

<b>Lógico</b>	Indica se o CGC informado é válido.
---------------	-------------------------------------

## **CONPAD1()**

---

A função CONPAD1() exibe uma tela de consulta padrão baseada no Dicionário de Dados (SXB).

- Sintaxe:** ConPad1 ( [uPar1] , [uPar2] , [uPar3] , cAlias , [cCampoRet] , [uPar4] , [IOnlyView] )

- Parâmetros:**

<b>uPar</b>	Parâmetro reservado.
<b>uPar2</b>	Parâmetro reservado.
<b>uPar3</b>	Parâmetro reservado.
<b>cAlias</b>	Consulta padrão cadastrada no Dicionário de Dados (SXB) a ser utilizada.
<b>cCampoRet</b>	Nome da variável ou campo que receberá o retorno da consulta padrão.
<b>uPar4</b>	Parâmetro Reservado.
<b>IOnlyView</b>	Indica se será somente para visualização.

- Retorno:**

<b>Nenhum</b>	.
---------------	---



*Anotações*

---

---

---

---

---

---

---

## **DATAVALIDA()**

A função DATAVALIDA() retorna a primeira data válida a partir de uma data especificada como referência, considerando inclusive a data informada para análise.

- Sintaxe: DATAVALIDA(dData)**

- Parâmetros:**

<b>dData</b>	Data a partir da qual será avaliada a próxima data válida, considerando-a inclusive como uma possibilidade.
--------------	---

- Retorno:**

<b>Data</b>	Próxima data válida, desconsiderando sábados, domingos e os feriados cadastrados no sistema.
-------------	--

## **EXISTINI()**

A função EXISTINI() verifica se o campo possui inicializador padrão.

- Sintaxe: EXISTINI(cCampo)**

- Parâmetros:**

<b>cCampo</b>	Nome do campo para verificação.
---------------	---------------------------------

- Retorno:**

<b>Lógico</b>	Indica se o campo possui um inicializador padrão.
---------------	---

### **Exemplo:**

```
// Exemplo de uso da função ExistIni:  
// Se existir inicializador no campo B1_COD:  
If ExistIni("B1_COD")  
    // Executa o inicializador:  
    cCod := CriaVar("B1_COD")  
Endif  
  
Return
```

## **EXTENSO()**

---

A função EXTERNO() retorna uma string referente a descrição por extenso de um valor numérico, sendo comumente utilizada para impressão de cheques, valor de duplicatas, etc.

- Sintaxe:** Extenso(**nValor**, **lQtd**, **nMoeda**)

- Parâmetros:**

<b>nValor</b>	Valor para geração do extenso.
<b>lQtd</b>	Indica se o valor representa uma quantidade (.T.) ou dinheiro (.F.)
<b>nMoeda</b>	Para qual moeda do sistema deve ser o extenso.

- Retorno:**

<b>String</b>	Descrição do valor por extenso.
---------------	---------------------------------

## **FORMULA()**

---

Interpreta uma fórmula cadastrada. Esta função interpreta uma fórmula, previamente cadastrada no Arquivo SM4 através do Módulo Configurador, e retorna o resultado com tipo de dado de acordo com a própria fórmula.

- Sintaxe:** Formula(**cFormula**)

- Parâmetros:**

<b>cFormula</b>	Código da fórmula a ser avaliada e cadastrada no SM4 – Cadastro de Fórmulas.
-----------------	--

- Retorno:**

<b>Indefinido</b>	Resultado da interpretação da fórmula cadastrada no SM4.
-------------------	--

## **GETADVFVAL()**

---

A função GETADVFVAL() executa uma pesquisa em um arquivo pela chave de busca e na ordem especificadas, possibilitando o retorno de um ou mais campos.

- Sintaxe:** GetAdvFVal(**cAlias**,**uCpo**,**uChave**,**nOrder**,**uDef**)

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo
<b>uCpo</b>	Nome de um campo ou array contendo os nomes dos campos desejados
<b>uChave</b>	Chave para a pesquisa
<b>nOrder</b>	Ordem do índice para a pesquisa
<b>uDef</b>	Valor ou array “default” para ser retornado caso a chave não seja encontrada

**Retorno:**

<b>Indefinido</b>	Retorna o conteúdo de um campo ou array com o conteúdo de vários campos
-------------------	---



A função GETADVVAL() difere da função POSICIONE() apenas por permitir o retorno de vários campos em uma única consulta.

As duas funções devem ser protegidas por GETAREA() / RESTAREA() dependendo da aplicação.

## **HELP()**

Esta função exibe a ajuda especificada para o campo e permite sua edição. Se for um help novo, escreve-se o texto em tempo de execução.

**Sintaxe: Help(cHelp,nLinha, cTitulo, uPar4,cMensagem,nLinMen,nColMen)**

**Parâmetros:**

<b>cHelp</b>	Nome da Rotina chamadora do help. (sempre branco)
<b>nLinha</b>	Número da linha da rotina chamadora. (sempre 1)
<b>cTitulo</b>	Título do help
<b>uPar4</b>	Sempre NIL
<b>cMensagem</b>	Mensagem a ser exibida para o Help.
<b>nLinMen</b>	Número de linhas da Mensagem. (relativa à janela)
<b>nColMen</b>	Número de colunas da Mensagem. (relativa à janela)

**Retorno:**

<b>Nenhum</b>
.



A função HELP() é tratada na execução das rotinas com o recurso de MSEXECAUTO(), permitindo a captura e exibição da mensagem no log de processamento.

Por esta razão, em rotinas que podem ser chamadas através da função MSEXECAUTO() deve-se sempre utilizar avisos utilizando esta função, para que este tipo de processamento não seja travado indevidamente.

**Exemplo:**

```
IF !Auto // Se for rotina automática
    Help("ESPECIFICO",1,"HELP","PROCESSAMENTO","Parâmetros do JOB Inválidos",1,0)
ELSE
    MsgBox("Parâmetros do JOB Inválidos", "PROCESSAMENTO")
ENDIF
```

## **MESEXTENSO()**

---

A função MESEXTENSO() retorna o nome de um mês por extenso.

- Sintaxe: MESEXTENSO(nMes)**

- Parâmetros:**

<b>nMes</b>	Indica o número do mês a ter seu nome escrito por extenso.
-------------	--

 Este parâmetro pode ser definido também como caractere ou como data.

*Importante*

- Retorno:**

<b>String</b>	Nome do mês indicado por extenso.
---------------	-----------------------------------

## **OBRIGATORIO()**

---

A função OBRIGATORIO() avalia se todos os campos obrigatórios de uma Enchoice() foram digitados.

- Sintaxe: OBRIGATORIO(aGets, aTela, aTitulos)**

- Parâmetros:**

<b>aGets</b>	Variável PRIVATE tratada pelo objeto Enchoice(), previamente definida no fonte.
<b>aTela</b>	Variável PRIVATE tratada pelo objeto Enchoice(), previamente definida no fonte.
<b>aTitulos</b>	Array contendo os títulos dos campos exibidos na Enchoice().

- Retorno:**

<b>Lógico</b>	Indica se todos os campos obrigatórios foram preenchidos.
---------------	---



*Anotações*

---

---

---

---

---

**Exemplo:**

```
#INCLUDE "protheus.ch"

/*
+-----+
| Programa | ATFA010A | Autor | ARNALDO R. JUNIOR | Data |
+-----+
| Desc.     | Cadastro de dados complementares do bem - Ativo Fixo |
+-----+
| Uso       | Curso de ADVPL |
+-----+
*/

User Function ATFA010A()

Private cCadastro := "Atualizacao de dados do bem"
Private aRotina := { {"Pesquisar" , "AxPesqui" , 0,1} ,
                    {"Visualizar" , "AxVisual" , 0,2} ,
                    {"Atualizar" , "U_A010AATU" , 0,4} }

Private cDelFunc := ".T."
Private cString := "SN1"

dbSelectArea("SN1")
dbSetOrder(1)
dbSelectArea(cString)
mBrowse( 6,1,22,75,cString)

Return

/*
+-----+
| Programa | A010AATU | Autor | ARNALDO R. JUNIOR | Data |
+-----+
| Desc.     | Atualização de dados do bem - Ativo Fixo |
+-----+
| Uso       | Curso de ADVPL |
+-----+
*/

User Function A010AATU(cAlias,nReg,nOpc)

Local aCpoEnch      := {}
Local aAlter         := {}

Local aButtons       := {}
Local cAliasE        := cAlias
Local aAlterEnch    := {}
Local aPos           := {015,000,400,600
Local nModelo        :=
Local lF3            := .F.
Local lMemoria       := .T.
Local lColumn        := .F.
Local caTela          := ""
Local lNoFolder      := .F.
Local lProperty       := .F.
```

**Exemplo (continuação) :**

```
Private oDlg
Private oGetD
Private oEnch
Private aTEL[0][0]      // Variáveis que serão atualizadas pela Enchoice()
Private aGETS[0]         // e utilizadas pela função OBRIGATORIO()

DbSelectArea("SX3")
DbSetOrder(1)
DbSeek(cAliasE)
//+-----+
//| Campos da enchoice
//+-----+
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
//+-----+
//| Campos alteráveis da enchoice
//+-----+
AADD(aAlterEnch, "N1_TIPOADT")      // Controle de Adiantamentos
AADD(aAlterEnch, "N1_DESCRIC")       // Descrição
AADD(aAlterEnch, "N1_CHAPA")        // Numero da placa
AADD(aAlterEnch, "N1_FORNEC")       // Fornecedor
AADD(aAlterEnch, "N1_LOJA")         // Loja do Fornecedor
AADD(aAlterEnch, "N1_NSERIE")       // Serie da Nota
AADD(aAlterEnch, "N1_NFISCAL")      // Numero da Nota
AADD(aAlterEnch, "N1_NFITEM")       // Item da Nota
AADD(aAlterEnch, "N1_UM")           // Unidade de Medida
AADD(aAlterEnch, "N1_PRODUTO")      // Código do Produto
AADD(aAlterEnch, "N1_PEDIDO")       // Código do Pedido de Compras
AADD(aAlterEnch, "N1_ITEMPED")      // Item do Pedido de Compras
AADD(aAlterEnch, "N1_PRCIMP")       // Código do Processo de Importação
AADD(aAlterEnch, "N1_CODPAIS")      // Código do País
AADD(aAlterEnch, "N1_ORIGCPR")      // Origem de Compras
AADD(aAlterEnch, "N1_CODSP")        // Código da SP Interna
AADD(aAlterEnch, "N1_CHASSIS")      // Numero de serie

//+-----+
//| Montagem do DIALOG
//+-----+
DEFINE MSDIALOG oDlg TITLE cCadastro FROM 000,000 TO 400,600 PIXEL
    RegToMemory("SN1", .F.)

    oEnch := MsMGet():New(cAliasE, nReg, nOpc, /*aCRA*/, /*cLetra*/;, 
/*cTexto*/;, aCpoEnch, aPos, aAlterEnch, nModelo, /*nColMens*/;, 
/*cMensagem*/;, /*cTudoOk*/;, oDlg, 1F3, lMemoria, lColumn,; 
caTela, lNoFolder, lProperty)

ACTIVATE MSDIALOG oDlg CENTERED;
ON INIT EnchoiceBar(oDlg, {||IIF(A010AGRV(aCpoEnch,aAlterEnch,nOpc),; 
oDlg:End(),.F.)},; // Botão OK
{||oDlg:End()},,aButtons) // Botão Cancelar

RETURN
```

**Exemplo (continuação) :**

```
/*
+-----+
| Programa | A010AGRV | Autor | ARNALDO R. JUNIOR | Data |
+-----+
| Desc.     | Validação da enchoice e gravação dos dados do bem |
+-----+
| Uso       | Curso de ADVPL |
+-----+
*/
Static Function A010AGRV(aCpos,aAlter,nOpc)

Local aArea      := GetArea()
Local nX         := 0

IF !Obrigatorio(aGets,aTela) /*Valida o cabecalho*/
    Return .F.
ENDIF

// Atualizacao dos campos passiveis de alteracao no SN1
RecLock("SN1",.F.)
For nX := 1 to Len(aAlter)
    SN1->&(aAlter[nX]) := M->&(aAlter[nX])
Next nX
MsUnLock()

Return .T.
```

---

## **OPENFILE()**

---

A função OPENFILE() exibe o diagnóstico de arquivos, verificando a existência dos arquivos de dados e os índices do sistema, criando caso não existam e abre os arquivos de acordo com o módulo onde é executada ou de acordo com a parametrização.

**Sintaxe: OPENFILE(cEmp)**

**Parâmetros:**

<b>cEmp</b>	Empresa cujo os arquivos serão re-abertos.
-------------	--

**Retorno:**

<b>Nenhum</b>	.
---------------	---

## **PERGUNTE()**

---

A função PERGUNTE() inicializa as variáveis de pergunta (mv\_par01,...) baseada na pergunta cadastrado no Dicionário de Dados (SX1). Se o parâmetro IAsk não for especificado ou for verdadeiro será exibida a tela para edição da pergunta e se o usuário confirmar as variáveis serão atualizadas e a pergunta no SX1 também será atualizada.

- Sintaxe:** Pergunte( cPergunta , [IAsk] , [cTitle] )

**Parâmetros:**

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

**Retorno:**

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

## **PESQPICT()**

---

A função PESQPICT() retorna a picture definida para um campo especificado no Dicionário de Dados (SX3).

- Sintaxe:** PesqPict(cAlias,cCampo,nTam)

**Parâmetros:**

<b>cAlias</b>	Alias do arquivo
<b>cCampo</b>	Nome do campo
<b>nTam</b>	Opcional, para campos numéricos, será usado como o tamanho do campo para definição da picture. Se não informado, é usado o tamanho padrão no dicionário de dados.

**Retorno:**

<b>String</b>	Picture do campo especificado.
---------------	--------------------------------



*Anotações*

---

---

---

---

---

---

---

## **PESQPICTQT()**

---

A função PESQPICTQT() retorna a picture de um campo numérico referente a uma quantidade, de acordo com o Dicionário de Dados (SX3). Esta função geralmente é utilizada quando há pouco espaço disponível para impressão de valores em relatórios, quando o valor nEdição não é informado, ela tem o comportamento semelhante ao da função "X3Picture", pois busca o tamanho do campo no dicionário de dados.

- Sintaxe:** PesqPictQt(cCampo,nEdição)

- Parâmetros:**

<b>cCampo</b>	Nome do campo a verificar a picture.
<b>nEdição</b>	Espaço disponível para edição.

- Retorno:**

<b>String</b>	Picture ideal para o espaço definido por nEdição, sem a separação dos milhares por vírgula.
---------------	---

## **POSICIONE()**

---

A função POSICIONE() permite o retorno do conteúdo de um campo de um registro de uma tabela especificado através de uma chave de busca.

- Sintaxe:** Posicione(cAlias, nOrdem, cChave, cCampo)

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo
<b>nOrdem</b>	Ordem utilizada
<b>cChave</b>	Chave pesquisa
<b>cCampo</b>	Campo a ser retornado

- Retorno:**

<b>Indefinido</b>	Conteúdo do campo solicitado.
-------------------	-------------------------------



*Importante*

A utilização da função POSICIONE() deve ser protegida com GETAREA() / RESTAREA() dependendo da aplicação.

## **PUTSX1()**

---

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

- Sintaxe:** PutSx1(cGrupo, cOrdem, cPergunt, cPerSpa, cPerEng, cVar, cTipo, nTamanho, nDecimal, nPresel, cGSC, cValid, cF3, cGrpSxg ,cPyme, cVar01, cDef01, cDefSpa1 , cDefEng1, cCnt01, cDef02, cDefSpa2, cDefEng2, cDef03, cDefSpa3, cDefEng3, cDef04, cDefSpa4, cDefEng4, cDef05, cDefSpa5, cDefEng5, aHelpPor, aHelpEng, aHelpSpa, cHelp)

- Parâmetros:**

<b>cGrupo</b>	Grupo de perguntas do SX1 (X1_GRUPO)
<b>cOrdem</b>	Ordem do parâmetro no grupo (X1_ORDEM)
<b>cPergunt</b>	Descrição da pergunta em português
<b>cPerSpa</b>	Descrição da pergunta em espanhol
<b>cPerEng</b>	Descrição da pergunta em inglês
<b>cVar</b>	Nome da variável de controle auxiliar (X1_VARIAVL)
<b>cTipo</b>	Tipo do parâmetro
<b>nTamanho</b>	Tamanho do conteúdo do parâmetro
<b>nDecimal</b>	Número de decimais para conteúdos numéricos
<b>nPresel</b>	Define qual opção do combo é o padrão para o parâmetro.
<b>cGSC</b>	Define se a pergunta será do tipo G – Get ou C – Choice (combo)
<b>cValid</b>	Expressão de validação do parâmetro
<b>cF3</b>	Código da consulta F3 vinculada ao parâmetro
<b>cGrpSxg</b>	Código do grupo de campos SXG para atualização automática, quando o grupo for alterado.
<b>cPyme</b>	Se a pergunta estará disponível no ambiente Pyme
<b>cVar01</b>	Nome da variável MV_PAR+ "Ordem" do parâmetro.
<b>cDef01</b>	Descrição da opção 1 do combo em português
<b>cDefSpa1</b>	Descrição da opção 1 do combo em espanhol
<b>cDefEng1</b>	Descrição da opção 1 do combo em inglês
<b>cCnt01</b>	Conteúdo padrão ou último conteúdo definido como respostas para este item
<b>cDef0x</b>	Descrição da opção X do combo em português
<b>cDefSpax</b>	Descrição da opção X do combo em espanhol
<b>cDefEngx</b>	Descrição da opção X do combo em inglês
<b>aHelpPor</b>	Vetor simples contendo as linhas de help em português para o parâmetro.
<b>aHelpEng</b>	Vetor simples contendo as linhas de help em inglês para o parâmetro.
<b>aHelpSpa</b>	Vetor simples contendo as linhas de help em espanhol para o parâmetro.
<b>cHelp</b>	Conteúdo do campo X1_HELP

## **RETINDEX()**

---

A função RETINDEX() restaura os índices padrões de um alias definidos no Dicionário de Dados (SIX).

- Sintaxe:** **RETINDEX(cAlias)**

- Parâmetros:**

<b>cAlias</b>	Alias de um arquivo do sistema existente no Dicionário de Dados.
---------------	--

- Retorno:**

<b>Numérico</b>	Indica quantos índices padrões o alias especificado possui no Dicionário de Dados.
-----------------	--



*Importante*

A função RETINDEX() quando utilizada em ambientes TOPCONNECT retorna -1

## **SIXDESCRICAO()**

---

A função SIXDESCRICAO() retorna a descrição da chave de índice, de acordo com o registro posicionado no SIX e idioma corrente.

- Sintaxe:** **SIXDESCRICAO()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>String</b>	Descrição do indice posicionado no SIX de acordo com o idioma corrente.
---------------	---

### **Exemplo:**

```
User Function <nome-da-função>( cChave, cOrdem )
Local cSixDesc := ""

dbSelectArea("SIX")
dbSetOrder(1)

If dbSeek(cChave+cOrdem)
    cSixDescr := SixDescricaو()
EndIf
Return
```

## **TABELA()**

---

A função TABELA() retorna o conteúdo de uma tabela cadastrada no Arquivo de Tabelas (SX5) de acordo com a chave especificada. Caso a tabela ou a chave especificada não existir será exibido um HELP() padrão do sistema.

- Sintaxe:** Tabela(cTab,cChav,lPrint)

- Parâmetros:**

<b>cTab</b>	Identificação da tabela a pesquisar (deve ser informado como caractere).
<b>cChav</b>	Chave a pesquisar na tabela informada.
<b>lPrint</b>	Indica se deve (.T.) ou não (.F.) exibir o help ou a chave NOTAB se a tabela não existir.

- Retorno:**

<b>String</b>	Conteúdo da tabela na chave especificada. Retorna nulo caso a tabela não exista ou a chave não seja encontrada.
---------------	---

## **TAMSX3()**

---

A função TAMSX3() retorna o tamanho (total e parte decimal) de um campo especificado no Dicionário de Dados (SX3).

- Sintaxe:** TAMSX3(cCampo)

- Parâmetros:**

<b>cCampo</b>	Nome do campo a ser consultado no Dicionário de Dados (SX3).
---------------	--

- Retorno:**

<b>Array</b>	Array de duas posições contendo o tamanho total e o número de decimais do campo especificado respectivamente.
--------------	---



*Anotações*

---

---

---

---

---

---

## **TM()**

A função TM() retorna a picture de impressão para valores numéricos dependendo do espaço disponível.

- Sintaxe: TM(nValor, nEdição, nDec)**

- Parâmetros:**

<b>nValor</b>	Valor a ser avaliado.
<b>nEdição</b>	Espaço disponível para edição.
<b>nDec</b>	Número de casas decimais.

- Retorno:**

<b>String</b>	Picture ideal para edição do valor nValor
---------------	---



*Importante*

Esta rotina leva em consideração duas variáveis:

- MV\_MILHAR – Determina se deve haver separação de milhar;
- MV\_CENT – Número de casas decimais padrão da moeda corrente.

Para ajustar o valor passado (nValor) ao espaço disponível (nEdição) a função verifica se pode haver separação de milhar, neste caso, a rotina eliminará tantos pontos decimais quantos sejam necessários ao ajuste do tamanho. Caso não seja possível ajustar o valor ao espaço dado, será colocado na picture o caracter de estouro de campo "\*". A função também ajusta um valor ao número de decimais (nDec), sempre imprimindo a quantidade de decimais passados no parâmetro.



*Anotações*

---

---

---

---

---

## **X1DEF01()**

A função X1DEF01() retorna o conteúdo da primeira definição da pergunta posicionada no SX1 (caso seja combo) no idioma corrente.

**Sintaxe: X1DEF01()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Conteúdo da primeira definição da pergunta no idioma corrente.
---------------	--

**Exemplo:**

```
User Function <nome-da-função>( cGrupo, cPerg )
Local cDef01
Local cDef02
Local cDef03
Local cDef04
Local cDef05

dbSelectArea("SX1")
dbSetOrder(1)

If dbSeek( cGrupo + cPerg ) // grupo da pergunta + o numero da perg.
    cDef01 := X1Def01()
    cDef02 := X1Def02()
    cDef03 := X1Def03()
    cDef04 := X1Def04()
    cDef05 := X1Def05()
EndIf

Return
```



*Anotações*

---

---

---

---

## **X1PERGUNT()**

A função X1PERGUNT() retorna a descrição da pergunta posicionada no Dicionário de Dados (SX1) para o idioma corrente.

**Sintaxe: X1PERGUNT()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Descrição da pergunta do Dicionário de Dados (SX1) no idioma corrente.
---------------	--

### **Exemplo:**

```
User Function <nome-da-função>( cGrupo, cPerg )
Local cDescr
dbSelectArea("SX1")
dbSetOrder(1)
If dbSeek( cGrupo + cPerg ) // grupo da pergunta + o numero da perg.
    cDescr := X1Pergunt()
EndIf
Return
```

## **X2NOME()**

A função X2NOME() retorna a descrição de uma tabela posicionada no Dicionário de Dados (SX2) no idioma corrente.

**Sintaxe: X2NOME()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Descrição da tabela posicionada no Dicionário de Dados (SX2) no idioma corrente.
---------------	--

### **Exemplo:**

```
User Function <nome-da-função>()
Local cTabela
dbSelectArea("SX2")
dbSetOrder(1)
If dbSeek( "SA1" )
    cTabela := X2Nome()
EndIf
Return
```

## **X3CBOX()**

A função X3CBOX() retorna o conteúdo de um campo tipo combo posicionado no Dicionário de Dados (SX3) no idioma corrente.

**Sintaxe: X3CBOX()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Conteúdo do campo do tipo combo posicionado no Dicionário de Dados (SX3) no idioma corrente.
---------------	--

### **Exemplo:**

```
User Function <nome-da-função>()
```

```
Local cTitulo  
Local cDescri  
Local cCombo
```

```
dbSelectArea("SX3")  
dbSetOrder(2)  
  
If dbSeek( cCampo )  
    cTitulo := X3Titulo()  
    cDescri := X3Descri()  
    cCombo := X3Cbox()  
EndIf  
  
Return
```

## **X3DESCRIC()**

A função X3DESCRIC() retorna a descrição de um campo posicionado no Dicionário de Dados (SX3) no idioma corrente.

**Sintaxe: X3DESCRIC()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Descrição do campo posicionado no Dicionário de Dados (SX3) no idioma corrente.
---------------	---

**Exemplo:**

```
User Function <nome-da-função>( )
```

```
Local cTitulo
```

```
Local cDescri
```

```
Local cCombo
```

```
dbSelectArea("SX3")
```

```
dbSetOrder(2)
```

```
If dbSeek( cCampo )
```

```
    cTitulo := X3Titulo()
```

```
    cDescri := X3Descri()
```

```
    cCombo := X3Cbox()
```

```
EndIf
```

```
Return
```

**X3PICTURE()**

A função X3PICTURE() retorna a máscara de um campo contido no Dicionário de Dados (SX3).

 **Sintaxe: X3PICTURE(cCampo)** **Parâmetros:**

<b>cCampo</b>	Nome do campo contido no Dicionário de Dados (SX3).
---------------	---

 **Retorno:**

<b>String</b>	Picture do campo informado.
---------------	-----------------------------

**Exemplo:**

```
User Function <nome-da-função>( cCampo )
```

```
Local cPicture
```

```
cPicture := X3Picture( cCampo )
```

```
Return cPicture
```



*Anotações*

---

---

---

---

---

## **X3TITULO()**

A função X3TITULO() retorna o título de um campo posicionado no Dicionário de Dados (SX3) no idioma corrente.

**Sintaxe: X3TITULO()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Título do campo posicionado no dicionário de dados (SX3) no idioma corrente.
---------------	--

### **Exemplo:**

```
User Function <nome-da-função>()
```

```
Local cTitulo
```

```
dbSelectArea("SX3")
dbSetOrder(2)
```

```
If dbSeek( "A1_COD" )
    cTitulo := X3Titulo()
EndIf
```

```
Return
```

## **X3USO()**

A função X3USO() verifica se o campo atualmente posicionado no Dicionário de Dados (SX3) está disponível para uso.

**Sintaxe: X3USO( cUsado, [Modulo] )**

**Parâmetros:**

<b>cUsado</b>	Conteúdo do campo X3_USADO a ser avaliado.
<b>Modulo</b>	Número do módulo. Caso não seja informado será assumido como padrão o número do módulo corrente.

**Retorno:**

<b>Lógico</b>	Indica se o campo está configurado como usado no Dicionário de Dados (SX3).
---------------	---

**Exemplo:**

```
User Function <nome-da-função>()

Local lUsado := .F.

DbSelectArea("SX3")
DbSetOrder(2)
DbSeek("A1_COD")

If X3Uso( SX3->X3_USADO )
    lUsado := .T.
EndIf

Return lUsado
```

**X5DESCRI()**

A função X5DESCRI() retorna a descrição de um item de uma tabela posicionado no Arquivo de Tabelas (SX5) no idioma corrente.

- Sintaxe: X5DESCRI()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>String</b>	Descrição do item do Arquivo de Tabelas (SX5) no idioma corrente.
---------------	---

**Exemplo:**

```
User Function <nome-da-função>( cFilial, cTabela, cChave )
Local cDescr

dbSelectArea("SX5")
dbSetOrder(1)

If dbSeek( cFilial+cTabela+cChave )
    cDescr := X5Descri()
EndIf

Return
```

## **X6CONTEUD()**

A função X6CONTEUD() retorna o conteúdo de um parâmetro posicionado no Dicionário de Dados (SX6) para o idioma corrente.

**Sintaxe: X6CONTEUD()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>Indefinido</b>	Conteúdo do parâmetro posicionado no Dicionário de Dados (SX6) para o idioma corrente.
-------------------	--



*Importante*

Utilizar preferencialmente as funções de manipulação de parâmetros GETMV() e suas variantes.

### **Exemplo:**

```
User Function <nome-da-função>( cFilial, cParam )
Local cDescr
Local cConceud

dbSelectArea("SX6")
dbSetOrder(1)

If dbSeek( cFilial+cParm )
    cDescr := X6Descric()
    cDescr += X6Desc1()
    cDescr += X6Desc2()
    cConceud := X6Conceud()
EndIf

Return
```



*Anotações*

---

---

---

---

## X6DESCRIC()

A função X6DESCRI() retorna o conteúdo da descrição de um parâmetro de acordo com o registro posicionado no Dicionário de Dados (SX6) no idioma corrente.

**Sintaxe: X6DESCRIC()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Descrição do parâmetro posicionado no Dicionário de Dados (SX6) no idioma corrente.
---------------	---



Dica

Para avaliar os conteúdos dos primeiro e segundo complementos da descrição do parâmetro utilize as funções:

- X6DESC01() → retorna o primeiro complemento da descrição.
- X6DESC02() → retorna o segundo complemento da descrição.

As três funções possuem a mesma sintaxe e forma de utilização.

### Exemplo:

```
User Function <nome-da-função>( cFilial, cParam )
```

```
Local cDescr  
Local cConteud
```

```
dbSelectArea("SX6")  
dbSetOrder(1)
```

```
If dbSeek( cFilial+cParm )  
    cDescr := X6Descric()  
    cDescr += X6Desc1()  
    cDescr += X6Desc2()  
    cConteud := X6Conteud()  
EndIf  
Return
```

## **XADESCRIC()**

A função XADESCRI() retorna o conteúdo da descrição dos folders de acordo com o registro posicionado no Dicionário de Dados (SXA) no idioma corrente.

**Sintaxe: XADESCRIC()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Descrição do folder posicionado no Dicionário de Dados (SXA) no idioma corrente.
---------------	--

### **Exemplo:**

```
User Function <nome-da-função>( cFolder, cNumero )
Local cDescr
dbSelectArea("SXA")
dbSetOrder(1)
If dbSeek( cFolder+cNúmero ) // alias do folder + numero do folder
    cDescr := XADescric()
EndIf
Return
```

## **XBDESCRI()**

A função XBDESCRI() retorna o conteúdo da descrição de uma consulta de acordo com o registro posicionado no Dicionário de Dados (SXB) no idioma corrente.

**Sintaxe: XBDESCRI()**

**Parâmetros:**

<b>Nenhum</b>	.
---------------	---

**Retorno:**

<b>String</b>	Descrição da consulta posicionada no Dicionário de Dados (SXB) no idioma corrente.
---------------	--

### **Exemplo:**

```
User Function <nome-da-função>( cAlias )
Local cDescr
dbSelectArea("SXB")
dbSetOrder(1)
If dbSeek( cAlias + "1" )
    cDescr := XBDescri()
EndIf
Return
```

## **XFILIAL()**

---

A função XFILIAL() retorna a filial utilizada por determinado arquivo.

Esta função é utilizada para permitir que pesquisas e consultas em arquivos trabalhem somente com os dados da filial corrente, dependendo é claro se o arquivo está compartilhado ou não (definição que é feita através do módulo Configurador – Dicionário de Dados (SX2)).

É importante verificar que esta função não tem por objetivo retornar apenas a filial corrente, mas retorná-la caso o arquivo seja exclusivo. Se o arquivo estiver compartilhado, a função xFilial retornará dois espaços em branco.

- Sintaxe:** XFILIAL(**cAlias**)

- Parâmetros:**

<b>cAlias</b>	Alias do arquivo desejado. Se não for especificado, o arquivo tratado será o da área corrente.
---------------	--

- Retorno:**

<b>Caracter</b>	String contendo a filial do arquivo corrente.
-----------------	---



*Anotações*

---

---

---

---

---

## Interfaces de cadastro

### AxCadastro()

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

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

**Sintaxe: AxCadastro(cAlias, cTitulo, cVldExc, cVldAlt)**

**Parâmetros:**

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

### Exemplo: Função AxCadastro()

```
#include "protheus.ch"

/*
+-----+
| Função | XCADSA2      | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Exemplo de utilização da função AxCadastro() |
|+-----+
| Uso       | Curso ADVPL |
|+-----+
*/

User Function XCadSA2()

Local cAlias      := "SA2"
Local cTitulo     := "Cadastro de Fornecedores"
Local cVldExc     := ".T."
Local cVldAlt     := ".T."

dbSelectArea(cAlias)
dbSetOrder(1)
AxCadastro(cAlias,cTitulo,cVldExc,cVldAlt)

Return Nil
```

## **Exemplo: Função de validação da alteração**

---

```
/*
+-----
| Função      | VLDALT          | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----
| Descrição  | Função de validação de alteração para a AXCADASTRO() |
+-----
| Uso        | Curso ADVPL           |
+-----
*/
User Function VldAlt(cAlias,nReg,nOpc)

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

nOpciao := AxAltera(cAlias,nReg,nOpc)

If nOpciao == 1
    MsgInfo("Ateração concluída com sucesso!")
Endif

RestArea(aArea)

Return lRet
```

---

## **Exemplo: Função de validação da exclusão**

---

```
/*
+-----
| Função      | VLDEXC          | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----
| Descrição  | Função de validação de exclusão para a AXCADASTRO() |
+-----
| Uso        | Curso ADVPL           |
+-----
*/
User Function VldExc(cAlias,nReg,nOpc)

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

nOpciao := AxExclui(cAlias,nReg,nOpc)

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

RestArea(aArea)
Return lRet
```

## **AXPESQUI()**

---

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

- Sintaxe: AXPESQUI()**
- Parâmetros**

<b>Nenhum</b>	.
---------------	---

## **AXVISUAL()**

---

Função de visualização padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

- Sintaxe: AXVISUAL(cAlias, nReg, nOpc, aAcho, nColMens, cMensagem, cFunc,; aButtons, lMaximized )**

- Parâmetros**

<b>cAlias</b>	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
<b>nReg</b>	Record number (recno) do registro posicionado no alias ativo.
<b>nOpc</b>	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização).
<b>aAcho</b>	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".
<b>nColMens</b>	Parâmetro não utilizado.
<b>cMensagem</b>	Parâmetro não utilizado.
<b>cFunc</b>	Função que deverá ser utilizada para carregar as variáveis que serão utilizadas pela Enchoice. Neste caso o parâmetro lVirtual é definido internamente pela AxFunction() executada como .T.
<b>aButtons</b>	Botões adicionais para a EnchoiceBar, no formato: aArray[n][1] -> Imagem do botão aArray[n][2] -> bloco de código contendo a ação do botão aArray[n][3] -> título do botão
<b>lMaximized</b>	Indica se a janela deverá ser ou não maximizada



### **Anotações**

---

---

---

---

---

## **AXINCLUI()**

---

Função de inclusão padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

- Sintaxe:** **AxInclui(cAlias, nReg, nOpc, aAcho, cFunc, aCpos, cTudoOk, IF3,; cTransact, aButtons, aParam, aAuto, lVirtual, lMaximized)**

**Parâmetros**

<b>cAlias</b>	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
<b>nReg</b>	Record number (recno) do registro posicionado no alias ativo.
<b>nOpc</b>	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização).
<b>aAcho</b>	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".
<b>cFunc</b>	Função que deverá ser utilizada para carregar as variáveis que serão utilizadas pela Enchoice. Neste caso o parâmetro lVirtual é definido internamente pela AxFunction() executada como .T.
<b>aCpos</b>	Vetor com nome dos campos que poderão ser editados
<b>cTudoOk</b>	Função de validação de confirmação da tela. Não deve ser passada como Bloco de Código, mas pode ser passada como uma lista de expressões, desde que a última ação efetue um retorno lógico: <b>"(Func1(), Func2(), ...,FuncX(), .T. )"</b>
<b>IF3</b>	Indica se a enchoice esta sendo criada em uma consulta F3 para utilizar variáveis de memória
<b>cTransact</b>	Função que será executada dentro da transação da AxFunction()
<b>aButtons</b>	Botões adicionais para a EnchoiceBar, no formato: aArray[n][1] -> Imagem do botão aArray[n][2] -> bloco de código contendo a ação do botão aArray[n][3] -> título do botão
<b>aParam</b>	Funções para execução em pontos pré-definidos da AxFunction(), conforme abaixo: aParam[1] := Bloco de código que será processado antes da exibição da interface. aParam[2] := Bloco de código para processamento na validação da confirmação. aParam[3] := Bloco de código que será executado dentro da transação da AxFunction(). aParam[4] := Bloco de código que será executado fora da transação da AxFunction().
<b>aAuto</b>	Array no formato utilizado pela funcionalidade MsExecAuto(). Caso seja informado este array, não será exibida a tela de interface, e será executada a função EnchAuto(). aAuto[n][1] := Nome do campo aAuto[n][2] := Conteúdo do campo aAuto[n][3] := Validação que será utilizada em substituição as validações do SX3

<b>IVirtual</b>	Indica se a Enchoice() chamada pela AxFunction() utilizará variáveis de memória ou os campos da tabela na edição
<b>IMaximized</b>	Indica se a janela deverá ser ou não maximizada

## **AXALTERA()**

---

Função de alteração padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

**Sintaxe: AXALTERA(cAlias, nReg, nOpc, aAcho, aCpos, nColMens, cMensagem,; cTudoOk, cTransact, cFunc, aButtons, aParam, aAuto, IVirtual, IMaximized)**

**Parâmetros**

➤ **Vide documentação de parâmetros da função AxInclui().**

## **AXDELETE()**

---

Função de exclusão padrão das informações de um registro, no formato Enchoice, conforme demonstrado no tópico sobre a interface AxCadastro().

**Sintaxe: AXDELETE(cAlias, nReg, nOpc, cTransact, aCpos, aButtons, aParam,; aAuto, IMaximized)**

**Parâmetros**

<b>cAlias</b>	Tabela cadastrada no Dicionário de Tabelas (SX2) que será editada
<b>nReg</b>	Record number (recno) do registro posicionado no alias ativo.
<b>nOpc</b>	Número da linha do aRotina que definirá o tipo de edição (Inclusão, Alteração, Exclusão, Visualização).
<b>cTransact</b>	Função que será executada dentro da transação da AxFunction()
<b>aCpos</b>	Vetor com nome dos campos que poderão ser editados
<b>aButtons</b>	Botões adicionais para a EnchoiceBar, no formato: aArray[n][1] -> Imagem do botão aArray[n][2] -> bloco de código contendo a ação do botão aArray[n][3] -> título do botão
<b>aParam</b>	Funções para execução em pontos pré-definidos da AxFunction(), conforme abaixo: aParam[1] := Bloco de código que será processado antes da exibição da interface. aParam[2] := Bloco de código para processamento na validação da confirmação. aParam[3] := Bloco de código que será executado dentro da transação da AxFunction(). aParam[4] := Bloco de código que será executado fora da transação da AxFunction().
<b>aAuto</b>	Array no formato utilizado pela funcionalidade MsExecAuto(). Caso seja informado este array, não será exibida a tela de interface, e será executada a função EnchAuto(). aAuto[n][1] := Nome do campo aAuto[n][2] := Conteúdo do campo aAuto[n][3] := Validação que será utilizada em substituição as

	validações do SX3
<b>IMaximized</b>	Indica se a janela deverá ser ou não maximizada

### **BRWLEGENDA()**

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

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

- Parâmetros:**

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



#### **Lista de cores disponíveis no Protheus**

- BR\_AMARELO
- BR\_AZUL
- BR\_BRANCO
- BR\_CINZA
- BR\_LARANJA
- BR\_MARRON
- BR\_VERDE
- BR\_VERMELHO
- BR\_PINK
- BR\_PRETO

### **ENDFILBRW()**

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

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

- Parâmetros:**

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

## **FILBROWSE()**

---

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

- Sintaxe: FilBrowse(cAlias, aQuery, cFiltro, IShowProc)**

- Parâmetros:**

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

## **PESQBRW()**

---

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

- Sintaxe: PesqBrw(cAlias , nReg, bBrwFilter)**

- Parâmetros:**

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



*Anotações*

---

---

---

---

---

## **MARKBROW()**

---

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

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

- Parâmetros:**

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

- Informações passadas para funções do aRotina:**

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

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

## Exemplo: Função MarkBrow() e acessórios

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

Local aCpos      := {}
Local aCampos    := {}
Local nI          := 0
Local cAlias      := "SA1"

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

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

AADD(aCpos, "A1_OK"      )
AADD(aCpos, "A1_FILIAL"  )
AADD(aCpos, "A1_COD"      )
AADD(aCpos, "A1_LOJA"     )
AADD(aCpos, "A1_NOME"     )
AADD(aCpos, "A1_TIPO"     )

dbSelectArea("SX3")
dbSetOrder(2)
For nI := 1 To Len(aCpos)
    IF dbSeek(aCpos[nI])
        AADD(aCampos, {X3_CAMPO, "", IIF(nI==1, "", Trim(X3_TITULO)), ;
                        Trim(X3_PICTURE) })
    ENDIF
Next

DbSelectArea(cAlias)
DbSetOrder(1)

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

Return Nil
```

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

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

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

Local nX := 0

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

RETURN
```



*Anotações*

---

---

---

---

## **MBROWSE()**

---

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

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

**Parâmetros:**

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

**Parâmetros:**

<b>cCpo</b>	Campo a ser validado se está vazio ou não para exibição do bitmap de status. Quando esse parâmetro é utilizado, a primeira coluna do browse será um bitmap indicando o status do registro, conforme as condições configuradas nos parâmetros <b>cCpo</b> , <b>cFun</b> e <b>aColors</b> .
<b>nPar08</b>	Parâmetro reservado.
<b>cFun</b>	Função que retornará um valor lógico para exibição do bitmap de status. Quando esse parâmetro é utilizado, o parâmetro <b>cCpo</b> é automaticamente desconsiderado.
<b>nClickDef</b>	Número da opção do aRotina que será executada quando for efetuado um duplo clique em um registro do browse. O default é executar a rotina de visualização.
<b>aColors</b>	Array bi-dimensional para possibilitar o uso de diferentes bitmaps de status. [n][1]=>Função que retornará um valor lógico para a exibição do bitmap [n][2]=>Nome do bitmap que será exibido quando a função retornar .T. (True). O nome do bitmap deve ser um resource do repositório e quando esse parâmetro é utilizado os parâmetros <b>cCpo</b> e <b>cFun</b> são automaticamente desconsiderados.
<b>cTopFun</b>	Função que retorna o limite superior do filtro baseado na chave de índice selecionada. Esse parâmetro deve ser utilizado em conjunto com o parâmetro <b>cBotFun</b> .
<b>cBotFun</b>	Função que retorna o limite inferior do filtro baseado na chave de índice selecionada. Esse parâmetro deve ser utilizado em conjunto com o parâmetro <b>cTopFun</b> .
<b>nPar14</b>	Parâmetro reservado.
<b>bInitBloc</b>	Bloco de código que será executado no ON INIT da janela do browse. O bloco de código receberá como parâmetro o objeto da janela do browse.
<b>INoMnuFilter</b>	Valor lógico que define se a opção de filtro será exibida no menu da MBrowse. .T. => Não exibe a opção no menu .F. => (default) Exibe a opção no menu. A opção de filtro na MBrowse está disponível apenas para TopConnect.
<b>ISeeAll</b>	Identifica se o Browse deverá mostrar todas as filiais. O valor default é .F. ( False ), não mostra todas as filiais. Caso os parâmetros <b>cTopFun</b> ou <b>cBotFun</b> sejam informados esse parâmetro será configurado automaticamente para .F. ( False ) Parâmetro válido à partir da <b>versão 8.11</b> . A função <b>SetBrwSeeAll</b> muda o valor default desse parâmetro.
<b>IChgAll</b>	Identifica se o registro de outra filial está autorizado para alterações. O valor default é .F. ( False ), não permite alterar registros de outras filiais. Quando esse parâmetro está configurado para .T. ( True ), o parâmetro <b>ISeeAll</b> é configurado automaticamente para .T. ( True ). Caso os parâmetros <b>cTopFun</b> ou <b>cBotFun</b> sejam informados esse parâmetro será configurado automaticamente para .F. ( False ). Parâmetro válido à partir da <b>versão 8.11</b> .

	A função <b>SetBrwChgAll</b> muda o valor default desse parâmetro.
--	--

**Variáveis private adicionais**

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

**Informações passadas para funções do aRotina:**

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

<b>cAlias</b>	Nome da área de trabalho definida para a Mbrowse
<b>nReg</b>	Recno do registro posicionado no Browse
<b>nOpc</b>	Posição da opção utilizada na Mbrowse de acordo com a ordem da função no array a Rotina.



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

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

## **Exemplo: Função Mbrowse()**

```
#include "protheus.ch"

/*
+-----+
| Função | MBRWSA1 | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Exemplo de utilização da função MBROWSE() |
+-----+
| Uso | Curso ADVPL |
+-----+
 */

User Function MBrwSA1()

Local cAlias := "SA1"
Private cCadastro := "Cadastro de Clientes"
Private aRotina := {}

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

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

Return Nil

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

#include "protheus.ch"

/*
+-----+
| Função | MBRWSA1 | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Exemplo de utilização da função MBROWSE() |
+-----+
| Uso | Curso ADVPL |
+-----+
 */

User Function MBrwSA1()

Local cAlias := "SA1"
Private cCadastro := "Cadastro de Clientes"
Private aRotina := {}

AADD(aRotina, {"Pesquisar", "AxPesqui", 0, 1})
AADD(aRotina, {"Visualizar", "AxVisual", 0, 2})
AADD(aRotina, {"Incluir", "U_Inclui", 0, 3})
```

**Exemplo (continuação):**

```
AADD(aRotina, {"Alterar", "AxAltera", 0, 4})
AADD(aRotina, {"Excluir", "AxDelete", 0, 5})

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

Return Nil
```

**// Exemplo: Função Inclui() substituindo a função AxInclui() - Função Inclui()**

```
/*
+-----+
| Função | INCLUI           | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Função de inclusão específica chamando a AXINCLUI()
+-----+
| Uso       | Curso ADVPL
+-----+
/*
```

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

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

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

If nOpcao == 1
    MsgBox("Inclusão concluída com sucesso!")
ElseIf      == 2
    MsgBox("Inclusão cancelada!")
Endif
```

```
Return Nil
```

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

```
#include "protheus.ch"

/*
+-----+
| Função | EXCLUI           | Autor | ARNALDO RAYMUNDO JR. | Data |
+-----+
| Descrição | Função de exclusão específica chamando a AxDelete
+-----+
| Uso       | Curso ADVPL
+-----+
/*
```

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

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

**Exemplo (continuação) :**

```
Local nOpcão          := 0
nOpcão := AxDelete(cAlias,nReg,aRotina[nOpc,4])
// Identifica corretamente a opção definida para o função em aRotinas com mais
// do que os 5 elementos padrões.

If nOpcão == 1
    MsgInfo("Exclusão realizada com sucesso!")
ElseIf      == 2
    MsgInfo("Exclusão cancelada!")
Endif

Return Nil
```

**Exemplo: Mbrowse() utilizando as funções acessórias**

```
#Include "Protheus.ch"

/*
+-----
| Programa | MBrwSA2   | Autor | SERGIO FUZINAKA | Data |
+-----
| Descrição | Exemplo da MBrowse utilizando a tabela de Cadastro de
|           | Fornecedores
+-----
| Uso       | Curso de ADVPL
+-----
*/
User Function MBrwSA2()

Local cAlias  := "SA2"
Local aCores  := {}
Local cFiltrar := "A2_FILIAL == '" +xFilial('SA2') + "' .And. A2_EST == 'SP'"

Private cCadastro := "Cadastro de Fornecedores"
Private aRotina  := {}
Private aIndexSA2 := {}
Private bFiltrarBrw:= { || FilBrowse(cAlias,@aIndexSA2,@cFiltrar) }

AADD(aRotina,{"Pesquisar"      , "PesqBrw"        ,0,1})
AADD(aRotina,{"Visualizar"    , "AxVisual"       ,0,2})
AADD(aRotina,{"Incluir"       , "U_BInclui"     ,0,3})
AADD(aRotina,{"Alterar"       , "U_BAltera"     ,0,4})
AADD(aRotina,{"Excluir"       , "U_BDelete"     ,0,5})
AADD(aRotina,{"Legenda"       , "U_BLegenda"    ,0,3})

/*
-- CORES DISPONIVEIS PARA LEGENDA --
BR_AMARELO
BR_AZUL
BR_BRANCO
BR_CINZA
BR_LARANJA
BR_MARRON
BR_VERDE
```

```

BR_VERMELHO
BR_PINK
BR_PRETO
*/
AADD(aCores, {"A2_TIPO == 'F'" , "BR_VERDE"      })
AADD(aCores, {"A2_TIPO == 'J'" , "BR_AMARELO"    })
AADD(aCores, {"A2_TIPO == 'X'" , "BR_LARANJA"   })
AADD(aCores, {"A2_TIPO == 'R'" , "BR_MARRON"     })
AADD(aCores, {"Empty(A2_TIPO)" , "BR_PRETO"      })

dbSelectArea(cAlias)
dbSetOrder(1)

//+-----
//| Cria o filtro na MBrowse utilizando a função FilBrowse
//+-----
Eval(bFiltrarBrw)

dbSelectArea(cAlias)
dbGoTop()
mBrowse(6,1,22,75,cAlias,,,,,aCores)

//+-----
//| Deleta o filtro utilizado na função FilBrowse
//+-----
EndFilBrw(cAlias,aIndexSA2)

Return Nil

//+-----
//| Função: BIInclui - Rotina de Inclusão
//+-----
User Function BIInclui(cAlias,nReg,nOpc)

Local nOpciao := 0

nOpciao := AxInclui(cAlias,nReg,nOpc)

If nOpciao == 1
    MsgInfo("Inclusão efetuada com sucesso!")
Else
    MsgInfo("Inclusão cancelada!")
Endif

Return Nil

```

**Exemplo (continuação) :**

```
//+-----  
//|Função: BALtera - Rotina de Alteração  
//+-----  
User Function BALtera(cAlias,nReg,nOpc)  
  
Local nOpcao := 0  
  
nOpcao := AxAltera(cAlias,nReg,nOpc)  
  
If nOpcao == 1  
    MsgInfo("Alteração efetuada com sucesso!")  
Else  
    MsgInfo("Alteração cancelada!")  
Endif  
  
Return Nil  
  
//+-----  
//|Função: BDelete - Rotina de Exclusão  
//+-----  
User Function BDelete(cAlias,nReg,nOpc)  
  
Local nOpcao := 0  
  
nOpcao := AxDelete(cAlias,nReg,nOpc)  
  
If nOpcao == 1  
    MsgInfo("Exclusão efetuada com sucesso!")  
Else  
    MsgInfo("Exclusão cancelada!")  
Endif  
  
Return Nil  
  
//+-----  
//|Função: BLegenda - Rotina de Legenda  
//+-----  
User Function BLegenda()  
  
Local aLegenda := {}  
  
AADD(aLegenda, {"BR_VERDE" , "Pessoa Física" })  
AADD(aLegenda, {"BR_AMARELO" , "Pessoa Jurídica" })  
AADD(aLegenda, {"BR_LARANJA" , "Exportação" })  
AADD(aLegenda, {"BR_MARRON" , "Fornecedor Rural" })  
AADD(aLegenda, {"BR_PRETO" , "Não Classificado" })  
  
BrwLegenda(cCadastro, "Legenda", aLegenda)  
  
Return Nil
```

## **MODELO2()**

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

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

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



*Importante*

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

- Sintaxe:** **Modelo2([cTitulo], [aCab], [aRoda], [aGrid], [nOpc], [cLinhaOk], [cTudoOk])**

- Parâmetros:**

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

**Retorno:**

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

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

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

USER FUNCTION MBrw2Sx5()

Local cAlias          := "SX5"

Private cCadastro      := "Arquivo de Tabelas"
Private aRotina        := {}
Private cDelFunc       := ".T." // Validacao para a exclusao. Pode-se utilizar
ExecBlock

AADD(aRotina, {"Pesquisar" , "AxPesqui" , 0,1})
AADD(aRotina, {"Visualizar" , "U_SX52Vis" , 0,2})
AADD(aRotina, {"Incluir"   , "U_SX52Inc" , 0,3})
AADD(aRotina, {"Alterar"   , "U_SX52Alt" , 0,4})
AADD(aRotina, {"Excluir"   , "U_SX52Exc" , 0,5})

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

Return

USER FUNCTION SX52INC(cAlias,nReg,nOpc)

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

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

**Exemplo (continuação):**

```
// Variaveis para campos da Enchoice()
Private cx5Filial := xFilial("SX5")
Private cx5Tabela := SPACE(5)

// Montagem do array de cabeçalho
// AADD(aCab, {"Variável" ,{L,C} , "Título", "Picture", "Valid", "F3", lEnable})
AADD(aCab, {"cx5Filial" ,{015,010} , "Filial", "@!,,,,.F."})
AADD(aCab, {"cx5Tabela" ,{015,080} , "Tabela", "@!,,,,.T."})

// Montagem do aHeader
AADD(aHeader, {"Chave" , "X5_CHAVE", "@!", 5, 0, "AllwaysTrue()", ;
    "", "C", "", "R"})
AADD(aHeader, {"Descricao" , "X5_DESCRI", "@!", 40, 0, "AllwaysTrue()", ;
    "", "C", "", "R"})

// Montagem do aCols
aCols := Array(1,Len(aHeader)+1)

// Inicialização do aCols
For nColuna := 1 to Len(aHeader)

    If aHeader[nColuna][8] == "C"
        aCols[1][nColuna] := SPACE(aHeader[nColuna][4])
    ElseIf aHeader[nColuna][8] == "N"
        aCols[1][nColuna] := 0
    ElseIf aHeader[nColuna][8] == "D"
        aCols[1][nColuna] := CTOD("")
    ElseIf aHeader[nColuna][8] == "L"
        aCols[1][nColuna] := .F.
    ElseIf aHeader[nColuna][8] == "M"
        aCols[1][nColuna] := ""
    Endif

Next nColuna

aCols[1][Len(aHeader)+1] := .F. // Linha não deletada
lRetMod2 := Modelo2(cTitulo, aCab, aRoda, aGrid, nOpc, cLinhaOk, cTudoOk)

IF lRetMod2
    //MsgInfo("Você confirmou a operação", "MBRW2SX5")
    For nLinha := 1 to len(aCols)
        // Campos de Cabeçalho
        Reclock("SX5", .T.)
        SX5->X5_FILIAL := cx5Filial
        SX5->X5_TABELA := cx5Tabela
        // Campos do aCols
        //SX5->X5_CHAVE := aCols[nLinha][1]
        //SX5->X5_DESCRI := aCols[nLinha][2]
        For nColuna := 1 to Len(aHeader)
            SX5->&(aHeader[nColuna][2]) := aCols[nLinha][nColuna]
        Next nColuna
        MsUnLock()
    Next nLinha
ELSE
    MsgAlert("Você cancelou a operação", "MBRW2SX5")
ENDIF
Return
```

## **MODELO3()**

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

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

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



*Importante*

- A função Modelo3() não implementa as regras de visualização, inclusão, alteração e exclusão, como uma AxCadastro() ou AxFunction().
- A inicialização dos campos utilizados na Enchoice() deve ser realizadas pela rotina que "suporta" a execução da Modelo3(), normalmente através do uso da função RegToMemory().
- Da mesma forma, o Browse deve ser tratado por esta rotina, sendo comum a Modelo3() estar vinculada ao uso de uma MBrowse().

**Sintaxe:** **Modelo3 ([cTitulo], [cAliasE], [cAliasGetD], [aCposE], [cLinOk], [cTudOk], [nOpcE], [nOpcG], [cFieldOk])**

**Parâmetros:**

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

**Retorno:**

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

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

```
#INCLUDE "protheus.ch"

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

User Function MbrwMod3()

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

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

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

Return

User Function Mod3All(cAlias,nReg,nOpcx)

Local cTitulo := "Cadastro de Pedidos de Venda"
Local cAliasE := "SC5"
Local cAliasG := "SC6"
Local cLinOk := "AlwaysTrue()"
Local cTudOk := "AlwaysTrue()"
Local cFieldOk:= "AlwaysTrue()"
Local aCposE  := {}
Local nUsado, nX  := 0
```

**Exemplo (continuação) :**

**Exemplo (continuação) :**

```
If Len(aCols)>0
    // Executa a Modelo 3
    // Executar processamento
    If lRetMod3 := Modelo3(cTitulo, cAliasE, cAliasG, aCposE, cLinOk, cTudOk,;
                           nOpcE, nOpcG,cFieldOk)
        // Executa a Modelo 3
        // Executar processamento
        If lRetMod3
            Aviso("Modelo3()", "Confirmada operacao!", {"Ok"})
        Endif
    Endif
Return
```

3

3

## Interfaces visuais para aplicações

### ALERT()

- Sintaxe: AVISO(cTexto)**
- Parâmetros**

cTexto	Texto a ser exibido
	



Anotações

---

---

---

---

- 529 -

## **AVISO()**

---

- Sintaxe:** AVISO(cTitulo, cTexto, aBotoes, nTamanho)
- Retorno:** numérico indicando o botão selecionado.
- Parâmetros:**

<b>cTitulo</b>	Título da janela
<b>cTexto</b>	Texto do aviso
<b>aBotoes</b>	Array simples (vetor) com os botões de opção
<b>nTamanho</b>	Tamanho (1,2 ou 3)
	

## **FORMBACTH()**

---

- Sintaxe:** FORMBATCH(cTitulo, aTexto, aBotoes, bValid, nAltura, nLargura )
- Parâmetros:**

<b>cTitulo</b>	Título da janela
<b>aTexto</b>	Array simples (vetor) contendo cada uma das linhas de texto que serão exibidas no corpo da tela.
<b>aBotoes</b>	Array com os botões do tipo SBUTTON(), com a seguinte estrutura:  <b>{nTipo,IEnable,{   Ação()}}</b>
<b>bValid</b>	(opcional) Bloco de validação do janela
<b>nAltura</b>	(opcional) Altura em pixels da janela
<b>nLargura</b>	(opcional) Largura em pixels da janela
	

## **MSGFUNCTIONS()**

---

- Sintaxe:** **MSGALERT(cTexto, cTitulo)**
- Sintaxe:** **MSGINFO(cTexto, cTitulo)**
- Sintaxe:** **MSGSTOP(cTexto, cTitulo)**
- Sintaxe:** **MSGYESNO(cTexto, cTitulo)**

### **Parâmetros**

<b>cTexto</b>	Texto a ser exibido como mensagem
<b>cTitulo</b>	Título da janela de mensagem
<b>MSGALERT</b>	
<b>MSGINFO</b>	
<b>MSGSTOP</b>	
<b>MSGYESNO</b>	

## Recursos das interfaces visuais

### GDFIELDGET()

A função GDFIELDGET() retorna o conteúdo de um campo especificado em uma grid formada por um objeto do tipo MsNewGetDados() de acordo com a linha da grid desejada.

- Sintaxe:** **GDFIELDGET(cCampo, nLinha)**

- Parâmetros:**

<b>cCampo</b>	Nome do campo para retorno do conteúdo.
<b>nLinha</b>	Linha da grid que deverá ser avaliada.

- Retorno:**

<b>Indefinido</b>	Conteúdo do campo especificado de acordo com a linha da grid informada.
-------------------	---

### GDFIELDPOS()

A função GDFIELDPOS() retorna a posição de um campo especificado em uma grid formada por um objeto do tipo MsNewGetDados().

- Sintaxe:** **GDFIELDPOS(cCampo)**

- Parâmetros:**

<b>cCampo</b>	Nome do campo a ser avaliado na grid.
---------------	---------------------------------------

- Retorno:**

<b>Numérico</b>	Posição que o campo ocupada na grid. Caso o mesmo não exista será retornado 0.
-----------------	--

### GDFIELDPUT()

A função GDFIELDPUT() atualiza o conteúdo de uma grid formada por um objeto do tipo MsNewGetDados() de acordo com o campo e linha da grid especificados.

- Sintaxe:** **GDFIELDPUT(cCampo, xConteudo, nLinha)**

- Parâmetros:**

<b>cCampo</b>	Nome do campo a ser atualizado.
<b>xConteudo</b>	Conteúdo que será atribuído a célula da grid.
<b>nLinha</b>	Linha da grid que será atualizada.

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **GETMARK()**

A função GETMARK() é utilizada em conjunto com a função MarkBrow(), para retornar o conjunto de caracteres que serão utilizados para identificar os registros marcados pelo browse.

- Sintaxe: GETMARK( [IUpper] )**

- Parâmetros:**

<b>IUpper</b>	Se verdadeiro (.T.) retorna somente caracteres em maiúsculos.
---------------	---

- Retorno:**

<b>String</b>	Conjunto de caracteres que definem a marca que deverá ser utilizada pela MarkBrowse durante o processamento corrente.
---------------	---



*Importante*

O retorno da função GETMARK() depende do conteúdo atual do parâmetro MV\_MARCA.



*Importante*

É altamente recomendável limpar o conteúdo do campo "marcado" pela MarkBrowse() ao término do processamento, para se evitar problemas com a reutilização da marca após a exaustão das possibilidades de combinação de dois caracteres, o qual é o tamanho padrão do campos utilizados para marcação de registros pela MarkBrowse(), que neste caso somam **1891** possibilidades de "00" a "zz".

### **Exemplo:**

```
Function <nome-da-função>()

Local aCampos := {{'CB_OK' ,,""},;
{'CB_USERLIB' ,,'Usuário'},;
{'CB_TABHORA' ,,'Hora'},;
{'CB_DTTAB' ,,'Data'}}

Private cMarca := GetMark()
Private cCadastro := 'Cadastro de Contrato'
Private aRotina := { { 'Pesquisar' , 'AxPesqui' , 0, 1 } }

MarkBrow( 'SCB', 'CB_OK','!CB_USERLIB',aCampos,, cMarca,'MarkAll()'...,'Mark()' )

Return
```

## **ISMARK()**

---

A função ISMARK() é utilizada em conjunto com a função MarkBrow() para identificar se o campo ou string contém a marca em uso no momento pela rotina.

- Sintaxe: ISMARK(cCampo, cMarca, lInvert)**

- Parâmetros:**

<b>cCampo</b>	Campo para ser avaliado em função da marca em uso pela MarkBrowse
<b>cMarca</b>	Marca em uso pela MarkBrowse
<b>lInvert</b>	Se foi utilizado o recurso de inverter seleção na MarkBrowse

- Retorno:**

<b>Lógico</b>	Se o conteúdo do campo ou string verificada contém a marca em uso pela MarkBrow().
---------------	--

## **MARKBREFRESH()**

---

A função MARKBREFRESH() atualiza a exibição da marca no MarkBrowse(), sendo utilizada quando algum processamento paralelo atualiza o conteúdo do campo definido como controle de marca para os registros em exibição pelo browse.

Este tipo de processamento é comum, e normalmente está associada a clique de "inverter" seleção, ou a opções de "marcar" e "desmarcar" todas.



*Importante*

A MarkBrowse() atualiza automaticamente a exibição da marca de registros quando utilizado o browse.

- Sintaxe: MARKBREFRESH()**

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Nenhum</b>	.
---------------	---

## **READVAR()**

---

A função READVAR() retorna o nome da variável ou campo associado ao objeto do tipo GET() atualmente selecionado ou em edição.

- Sintaxe:** READVAR()

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>String</b>	Nome da variável ou campo associado ao objeto do tipo GET.
---------------	--

## **THISINV()**

---

A função THISINV() é utilizada em conjunto com a função MarkBrow() para identificar se o foi utilizado o recurso de inversão de seleção dos itens exibidos em uma MarkBrowse.

- Sintaxe:** THISINV()

- Parâmetros:**

<b>Nenhum</b>	-
---------------	---

- Retorno:**

<b>Lógico</b>	Indica se foi utilizada a opção de inversão da seleção dos itens exibidos em uma MarkBrowse.
---------------	--

## **THISMARK()**

---

A função THISMARK() é utilizada em conjunto com a função MarkBrow(), e retorna a marca em uso no momento para a MarkBrow().

- Sintaxe:** THISMARK()

- Parâmetros:**

<b>Nenhum</b>	.
---------------	---

- Retorno:**

<b>Caracter</b>	Marca em uso pela MarkBrow().
-----------------	-------------------------------