ADVPL I

Aprendendo a Linguagem

Conteúdo Programático

Conteúdo Programático

- A LINGUAGEM ADVPL
- PROGRAMAÇÃO (ESTRUTURA, DOCUMENTAÇÃO, FUNCIONALIDADES)
- OPERADORES
- MACROSUBSTITUIÇÃO
- VARIÁVEIS (TIPOS , UTILIDADES, PASSAGEM DE PARÂMETROS)
- MATRIZES (TEORIA e EXERCÍCIOS)
- CONTROLE DE FLUXO
- ASSISTENTE DE CÓDIGO (CADASTROS , RELATORIOS E TEXTO)

Conteúdo Programático

- WIZARD (INICIALIZAÇÃO E CONFIGURAÇÃO DO ARQUIVO INI)
- CONFIGURADOR
- APRENDENDO O IDE (AMBIENTE DE DESENVOLVIMENTO)
- CAIXAS DE MENSAGENS
- DESENVOLVENDO A LINGUAGEM COM MATRIZES
- BLOCO DE CÓDIGO
- RELACIONAMENTO ENTRE TABELAS

Conteúdo Programático

- ►FUNÇÕES DE BASE DE DADOS
- ► PONTO DE ENTRADA
- ► NORTON GUIDE
- ► MPSDU / APSDU
- ► FUNÇÕES DIVERSAS
- ►TELAS PADRAO MS
- ►ROTINAS AUTOMATICAS

Origem da ADVPL

Origem da ADVPL

- Surgiu com a tecnologia Protheus em 1994
- Derivada do Clipper
- Linguagem orientada a objetos
- Ambiente de desenvolvimento (IDE)
- APO´s (Advanced Protheus Objects)

Origem da ADVPL

✓ **Características**

- Interface com o Usuário
- Processos
- Programação de RPC
- Programação Web
- Programação TelNet

Como Programar

Como Programar

- ✓ **PROGRAMAÇÃO**
  - ➢ Compilação
  - ➢ Execução

Como Programar

- ✓ **Linhas de Programa**
  - ➢ Linha de comentário
  - ➢ Linha de comando
  - ➢ Linha mista
  - ➢ Linha lógica
  - ➢ Linha física

Como Programar

- ✓ **Linhas de Programa**
  - ➢ Linhas de comentário

*** (asterisco)**	*Programa para recalculo do custo médio
&&	&&Programa para recalculo do custo médio
//	// Programa para recalculo do custo médio

Como Programar

- ✓ **Comentário em bloco**

  - ➢ Podemos documentar blocos de texto inicializando com /* e finalizando com */

```
/*
Programa para recalculo do custo médio
Autor: Biale ADVPL e Consultoria em Informática
Data: 15 de dezembro de 2006
*/
```

Como Programar

- ✓ Linhas de comando
  - ➢ Local nSoma := 0
- ✓ Linhas mistas
  - ➢ Local nSoma := 0 //Utilizada para totalizar o relatório.

Como Programar

Como Programar

- **Tamanho de Linha**
  - Linha física
  - Linha lógica

```
If !Empty(cNome) .And. !Empty(cEnd) .And. ;
   !Empty(cTel)  .And. !Empty(cFax) .And. ;
   !Empty(cEmail)
   GravaDados(cNome,cEnd,cTel,cFax,cEmail)
Endif
```

Estruturação

- ✓ **Identificação**

```
/*****************************************************************
*******\
|Funçao para separação de números pares e impares|
|Autor: Cristiane C. Figueiredo|
|Data : 01/03/06|
\*****************************************************************
*******/
```

Estruturação

- ✓ **Inicialização**

```
User Function fSeparaNum
Local nBI
Local cBIImpares := ""  //Numeros Impares
Local cBIPares   := ""   //Números Pares
```

Estruturação

- ✓ **Corpo do programa**

```
For nBI:=1 to 12
        If mod(Nbi,2)== 0
                cBIPares   += Alltrim(str(nBI)) + " "
        Else
                cBIImpares += Alltrim(str(nBI)) + " "
        Endif
Next
```

Estruturação

- ✓ **Encerramento**

```
// Exibe em tela o resultado encontrado
Msginfo("Pares “ + cBIPares + " Impares “ + cBIImpares)
// Termina o programa
Return
```

Documentações

```
/*************************************************************\
|Funcao para separação de números pares e impares|
|Autor: Cristiane C. Figueiredo|
|Data : 01/03/06|
\*************************************************************/
User Function fSeparaNum
Local nBI
Local cBIImpares := “” //Numeros Impares
Local cBIPares   := ""  //Números Pares
For nBI:=1 to 12
        If mod(nBI,2)== 0
            cBIPares   += Alltrim(str(nBI)) + “ “
        Else
            cBIImpares += Alltrim(str(nBI)) + “ “
        Endif
Next
// Exibe em tela o resultado encontrado
Msginfo("Pares “ + cBIPares + " Impares “ + cBIImpares)
Return
```

Documentações

- ✓ **Funções**

```
/*=============================================================================|
|Função    |ADV0070A       |Autor|Cristiane C. Figueiredo     | Data |23/07/04|
|==========|=============================================================|
|Descrição |Verifica se campo do E2 pode ser alterado ou não               |
|==========|=============================================================|
|Modulo    | SIGAFIN                                                       |
|==========|=============================================================|
|OBS       | Especifico para BIALE                                         |
|==========|=============================================================|
|Alterações solicitadas                                                    |
|=============================================================================|
|Descrição                                      |Sol.por| Atend.por  |Data    |
|=============================================================================|
|Incluida verificação de chamada por MsExecAuto                            |
|ISInCallStack('msexecauto‘)                    |Gerente|Lucas Borges|30/01/07|
|=============================================================================*/
User Function ADV0070A(nCampo)
       Comandos...
Return .T.\
```

Documentações

- ✓ **Solicitação de Customizações**

Dados da Requisição

Data Solicitação	data da solicitação
Área	nome da área solicitante/centro de custo
Responsável	nome do solicitante responsável
Analista de negócio	nome do analista de negócio
Volume de esforço previsto	quantidade de horas necessárias para o desenvolvimento/testes
Data de entrega prevista	data de entrega prevista (uso da área de TI)

Documentações

- ✓ **Aprovações**

Nome	Assinatura	Data

Documentações

- ✓ Resumo
  - ➢ Objetivo
  - ➢ Programas envolvidos (análise inicial)
  - ➢ Comentários/Observações do Responsável pela Customização

Funcionalidades

✓ **Tipos de Funções**

Function	Limitada a programas padrões, não permitindo ser compilada por clientes.
User Function	Funções de usuário, podendo ser utilizadas para fazer customizações.
Static Function	Funções chamadas pelas User Functions e pelas Functions.

Variáveis

- ✓ **Escopos de variáveis**

Public	Visível a partir do momento que é criada até o termino da aplicação, não deve ser utilizada para qualquer coisa pois ocupa espaço de memória.
Private	Visível na função que a criou e nas demais chamadas por ela, até que a função que a criou seja finalizada e são criadas automaticamente quando não declaradas.
Local	Visível somente na função que a criou, variáveis locais ocupam pouco espaço de memória e são criadas automaticamente quando provierem de parametrização de função

Variáveis

- ✓ **Tipos de Variáveis**

Tipo	Exemplo	Descritivo
Caracter	cVar	Aceita números e letras
Numérica	nVar	Aceita apenas números
Data	dVar	Aceita apenas Datas
Lógica	lVar	Aceita Verdadeiro ou Falso
Array	aVar	Aceita Vetores
Objeto	oVar	Usada em objetos
Bloco	bVar	Bloco de Código

Variáveis

- ✓ Variáveis e nomes de campos

```
cRes := MEMVAR->NOME
cRes := FIELD->NOME
```

- ✓ Inicialização de Variáveis

Como Programar

CRIAÇÃO DE VARIÁVEIS

✓ **Utilização da Função CRIAVAR( )**

Inicializa variáveis obedecendo o dicionário de dados.

Sintaxe: uRet := CriaVar(cCampo,lIniPad,cLado)

Argumento	Obrigat.	Tipo	Descrição
uRet	Sim	C	tipo de retorno de acordo com o dicionário de dados.
cCampo	Sim	N	Nome do campo
lIniPad	Não	L	Indica se considera (.T.) ou não (.F.) o inicializador padrao (X3_RELACAO)
cLado	Não	C	Se campo for caracter inicializa alinhado a esquerda (L), direita(D) ou centralizado(C).

Como Programar

OPERADORES BÁSICOS

- ✓ **Matemáticos**

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
** ou ^	Exponenciação
%	Módulo (Resto da Divisão)

OPERADORES BÁSICOS

- ✓ **String**

+	Concatenação de strings (união)
$	Comparação de Substrings (está contido em)

Como Programar

OPERADORES BÁSICOS

- ✓ **Relacionais**

<	Menor que
>	Maior que
=	Igual
==	Exatamente Igual (para caracteres)
<=	Menor ou Igual
>=	Maior ou Igual
<> ou # ou !=	Diferente

Como Programar

OPERADORES BÁSICOS

- ✓ **Lógicos**

.And.	E
.Or.	OU
.Not. ou !	NÃO

Como Programar

OPERADORES BÁSICOS

- ✓ **Atribuições**

:=	Atribuição
+=	Adiciona e depois atribui
-=	Subtrai e depois atribui
*=	Multiplica e depois atribui
/=	Divide e depois atribui
++	Incremento Pós ou Pré-fixado
--	Decremento Pós ou Pré-fixado

Como Programar

OPERADORES BÁSICOS

- ✓ **Especiais**

( )	Agrupa elementos em uma expressão mudando a **ordem de precedência** de avaliação da expressão. Também servem para envolver os argumentos de uma função.
[ ]	Elemento de Matriz. Utilizados para especificar um elemento específico de uma matriz. Ex. aArray[1]
{}	Definição de Matriz, Constante ou Bloco de Código. Por exemplo: aArray := {1,2,3,4,5} cria uma matriz chamada aArray com cinco elementos.

Como Programar

OPERADORES BÁSICOS

- ✓ **Especiais**

->	Identificador de Apelido (Alias). identifica um campo de um arquivo diferenciando-o de uma variável. Exemplo: SA1->A1_NOME
&	Identifica uma avaliação de expressão através macrossubstituição.

Como Programar

PRECEDÊNCIA

- ✓ Incremento/Decremento pré-fixado
- ✓ String
- ✓ Matemáticos
- ✓ Relacionais
- ✓ Lógicos
- ✓ Atribuição
- ✓ Incremento/Decremento pós-fixado

PRECEDÊNCIA

Matemática

- ✓ Exponenciação
- ✓ Multiplicação e Divisão
- ✓ Adição e Subtração

Como Programar

PRECEDÊNCIA

Exemplo

- ✓ Local nVar := 15+12/3+7*3-2^3
- ✓ Calcula-se a exponenciação: 2^3 =8
- ✓ Calcula-se divisão: 12/3 =4
- ✓ Calcula-se a multiplicação: 7*3 =21
- ✓ Efetua a soma e subtração: 15 + 4 + 21 - 8
- ✓ O resultado desta expressão é 32

Como Programar

PRECEDÊNCIA

Exemplo

- ✓ Local nVar := (15+12)/(3+7)*3-2^3
- ✓ Calcula-se a exponenciação: 2^3 =8
- ✓ Calcula-se o primeiro grupo de parênteses: 15+12 = 27
- ✓ Calcula-se o segundo grupo: 3+7=10
- ✓ Calcula-se a divisão: 27/10 = 2.7
- ✓ Calcula-se a multiplicação: 2.7 * 3 = 8.1
- ✓ Efetua a soma e subtração: 8.1-8
- ✓ O resultado desta expressão é 0.1

Como Programar

MACROSUBSTITUIÇÃO

Exemplo

- ✓ nVar := 20
- ✓ cVar := “nVar + 35”
- ✓ nResult := &cVar
- ✓ nResult := nVar + 35
- ✓ nResult := 20 + 35
- ✓ nResult := 55

- ✓ ***Exercícios da página 18***

Vetores

Utilizando Vetores

- ✓ **Exemplo**

```
//Declaração da variável
Local aArray

//Atribuição da matriz à variável
aArray := {{"Pedro", "Louro", "Masculino"}, ;
           {"João", "Moreno", "Masculino"}, ;
           {"Maria", "Ruiva", "Feminino"}}
```

Utilizando Vetores

✓ **Exemplo**

```
MsgInfo(aArray[nLin][nCol])

Onde:

nLin -> Linha onde se encontra o conteúdo que se quer exibir
nCol -> Coluna onde se encontra o conteúdo que se quer exibir

MsgInfo(aArray[3][2])

Resposta: "Ruiva"
```

Utilizando Vetores

✓ **Exemplo**

```
aArray[nLin][nCol] := cVar

Onde:

nLin -> Linha onde se encontra o conteúdo que se quer alterar
nCol -> Coluna onde se encontra o conteúdo que se quer alterar
cVar -> Novo Conteúdo

aArray[3][2] := "Mulata"
```

Utilizando Vetores

- ✓ **Cuidados**

  O tamanho do array, pode comprometer a memória do servidor.

- ✓ **Estrutura**

  O array pode ser construído com uma estrutura mista, ou seja,
  Aceita números, caracteres, data, inclusive um outro array.

Utilizando Vetores

- ✓ ***Exercícios da página 21***

Utilizando Vetores

PAG. 21

RESPOSTA EXERCICIO 1

Criar um vetor de 5 linhas com 3 Colunas e dar o nome de aArray1

```
aArray1:= {{"Maria",15,"Gato" };
           {"João",60,"Papagaio"};
           {"Pedro",45,"Cachorro"};
           {"Vitor",1,"Cadela"};
           {"Bianca",8,"Coelho"}}
```

Utilizando Vetores

RESPOSTA EXERCICIO 2

Criar um vetor de 10 linhas com 1 Colunas e dar o nome de aArray2

```
aArray2 := {{"Cachorro"}, {"Papagaio"},{"Gato"},;
            {"Macaco"},{"Cavalo"},{"Pato"},{"Tatu"},;
            {"Urubu"},{"Cisne"},{"Rato"}}
```

Utilizando Vetores

RESPOSTA EXERCICIO 3

Atribuir ao Vetor a Array1 linha 1 coluna 1 o conteúdo "X"

```
aArray1[1][1] := "X"
```

Utilizando Vetores

RESPOSTA EXERCICIO 4

Atribuir ao Vetor a Array1 linha 3 coluna 2 o conteúdo 9

```
aArray1[3][2] := "9"
```

Utilizando Vetores

RESPOSTA EXERCICIO 5

Atribuir ao Vetor a Array1 linha 5 coluna 3 o conteúdo .T.

```
aArray1 [5][3] := .T.
```

Utilizando Vetores

RESPOSTA EXERCICIO 6

Atribuir ao Vetor a Array1 linha 1 coluna 3 o conteúdo do aArray2 Linha 5 coluna 1.

```
aArray1 [1][3] := aArray2 [5][1]
```

Funções de Array

Utilizando Vetores

Aadd()
Adiciona um novo elemento ao final do array.

```
aArray := {}                        // Resultado: aArray
vazio
AADD(aArray, 5)              // Resultado: { 5 }
AADD(aArray, 10)             // Resultado: { 5, 10 }
AADD(aArray, { 12, 10 })          // Resultado: { 5, 10, {
12, 10 } }

aArray[1]    // Resultado 5
aArray[2]    // Resultado 10
aArray[3][1]         // Resultado 12
```

Utilizando Vetores

Funções

Array()

Cria um array com dados não inicializados.

```
aArray := ARRAY(5)
aArray := { NIL, NIL, NIL, NIL, NIL }

aArray := ARRAY(3, 2)
aArray := { {NIL, NIL}, {NIL, NIL}, {NIL, NIL} }
```

Utilizando Vetores

Funções

Ascan()

Faz uma busca em um array.

```
aArray := { “Antonio”, "Maria", "Sueli" }
nPos := ASCAN(aArray, “Maria") // Resultado: 2
nPos := ASCAN(aArray, “maria") // Resultado: 0

AADD(aArr,{“Paulo",“Paulista"})
AADD(aArr,{“Pedro",“Carioca"})
AADD(aArr,{“Mariana",“Paranaense"})
AADD(aArr,{“Patricia",“Gaucha"})
nPos := ASCAN(aArr, {|x| x[1] == “Mariana"})
//Resultado: 3
nPos := ASCAN(aArr, {|x| x[2] == “Carioca"})
//Resultado: 2
```

Utilizando Vetores

Funções

Asort()

Ordena um array.

```
aArray := { 3, 5, 1, 2, 4 }
aArray := ASORT(aArray) // Resultado: { 1, 2, 3, 4, 5 }

aArray := {{"Maria",14},{"Joao",23},{"Aline",16},{"Eva",35}}

Ordem Crescente pelo 2° elemento
aArray := ASORT(aArray ,nIni,nFim, { |x, y| x[2] < y[2] })
Resultado: {{"Maria",14},{"Aline",16},{"Joao",23},{"Eva",35}}

Ordem Decrescente pelo 2° elemento
aArray := ASORT(aArray ,,, { |x, y| x[2] > y[2] })
Resultado:{{"Eva",35},{"Joao",23},{"Aline",16},{"Maria",14}}

Ordem Crescente pelo 1° elemento
aArray := ASORT(aArray ,,, { |x, y| x[1] < y[1] })
Resultado:{{"Aline",16},{"Eva",35},{"Joao",23},{"Maria",14}}
```

Controle de fluxo

Controle de Fluxo

- ✓ Estruturas de Decisão
- ✓ Estruturas de Repetição

Controle de Fluxo

Estrutura de repetição

For ... Next

Sintaxe

FOR *Variavel := nValorInicial* TO *nValorFinal [*STEP *nIncremento]*

Comandos...

[EXIT]

[LOOP]

NEXT

Controle de Fluxo

Estrutura de repetição

Exemplo:

```
Local cBIPares    := “”
Local cBIImpares  := “”
For nBI:=1 to 12
    If mod(nBI,2)== 0
      cBIPares   += cValtoChar(nBI) + “ ”
    Else
      cBIImpares += cValtoChar(nBI) + “ ”
    Endif
Next
MsgInfo(“Numeros Pares: ” + cBIPares)
MsgInfo(“Numeros Impares: ” + cBIImpares)
```

Controle de Fluxo

Estrutura de repetição

While Enddo

Sintaxe

WHILE lExpressao

Comandos...

[EXIT]

[LOOP]

ENDDO

Controle de Fluxo

Estrutura de repetição

Exemplo:

```
Local nSomaPar := 0
Local nNumber  := 350

While nNumber > 0
     nSomaPar++
     nNumber -= 2
Enddo

Alert("Quantidade No. Pares: "+cValToChar(nSomaPar))
```

Controle de Fluxo

Estrutura de decisão

If ... Endif

Sintaxe

IF lExpressao

Comando Verdadeiro...

ELSE

Comando Falso...

ENDIF

Controle de Fluxo

Estrutura de decisão

Exemplo:

```
Local dVencto := CTOD("31/12/01")
     If Date() > dVencto
       Alert("Titulo vencido!")
     Else
       Alert("Titulo a vencer!")
     Endif
Return
```

Controle de Fluxo

Estrutura de Decisão

Do Case ... EndCase

Sintaxe

```
DO CASE
        CASE lExpressao1
            Comandos
        CASE lExpressao2
            Comandos
            ...
        CASE lExpressaoN
            Comandos
        [OTHERWISE]
            Comandos
ENDCASE
```

Controle de Fluxo

Estrutura de Decisão

Exemplo:

```
Local nMes      := Month(Date())
Local cPeriodo := ""
DO CASE
     CASE nMes <= 3
       cPeriodo := "Primeiro Trimestre"
     CASE nMes <= 6
       cPeriodo := "Segundo Trimestre"
     CASE nMes <= 9
       cPeriodo := "Terceiro Trimestre"
     OTHERWISE
       cPeriodo := "Quarto Trimestre"
ENDCASE
Return
```

Controle de Fluxo

Problemas Comus

Exemplo 1:

```
dbSeek(xFilial(“SE1”)+DTOS(dDtIni))

Do While SE1->(!Eof())

        @ lin, col SAY “teste”

        SA1->(DBSKIP())

        SE1->(DBSKIP())

Enddo
```

Controle de Fluxo

Problemas Comuns

Exemplo 2:

```
aCampos := {}
Do while .T.
        Aadd(aCampos, “Teste”)
        If Len(aCampos) > 100
                EXIT
        Endif
Enddo
```

Controle de Fluxo

Problemas Comuns

Exemplo 3:

```
aCampos := {}
dbSeek(xFilial("SE1")+DTOS(dDtIni))
Do While SE1->(!Eof())
        If !EMPTY(SE1->E1_BAIXA)
                Reclock("SE5"      , .T.)
        Endif
        SE1->(dbSkip())
Enddo
```

Resposta dos Exercícios

Resposta do exercicio 1

Usando For…Next

```
User Function fExFluxo1
Local aArray := { { "Linha1", "A", "B", "C", "D" },;
                  { "Linha2", "E", "F", "G", "H" },
                  { "Linha3", "I", "J", "K", "L" }}
      For I := 1 to 3
            Mostre na tela aArray[I][1]
            Mostre na tela aArray[I][2]
            Mostre na tela aArray[I][3]
            Mostre na tela aArray[I][4]
            Mostre na tela aArray[I][5]
      Next
Return
```

Resposta dos Exercícios

Resposta do exercicio 1

Usando Len()

```
User Function fExFluxo1()
Local aArray := { { "Linha1", "A", "B", "C", "D" },;
                  { "Linha2", "E", "F", "G", "H" },;
                  { "Linha3", "I", "J", "K", "L" }}
      I := 1
      While I <= len(aArray)
            Mostre na tela aArray[I][1]
            Mostre na tela aArray[I][2]
            Mostre na tela aArray[I][3]
            Mostre na tela aArray[I][4]
            Mostre na tela aArray[I][5]
            I++
      Enddo
Return
```

Resposta dos Exercícios

Resposta do exercicio 2

Usando For…Next

```
User Function fExFluxo2()
Local aArray := { { "Maria", 10, 7, 15, 31} ,;
                  { "Jose ", 15, 16, 21, 33} , ;
                  { "Petrucio", 8, 8, 8, 6} , ;
                  { "Firmino", 15, 16, 21, 33} , ;
                  { "Felizberto", 10, 17, 31, 25} }
   For I:=1 to Len(aArray)
      Mostra "Nome " + aArray[I][1]
      nMedia := (aArray[I][2] + aArray[I][3] +;
                 aArray[I][4] + aArray[I][5])/4
      Mostra "Media " + str(nMedia)
   Next
Return
```

Resposta dos Exercícios

Usando If… Else…Endif

```
User Function fExFluxo3()
Local aArr:={{“Maria”,10,7,15,31},;
             {“Jose”,15,16,21,33},;
             {“Petrucio”,8,8,8,6},;
             {“Firmino”,15,16,21,33},;
             {“Felizberto”, 10, 17, 31, 25} }
   For I:=1 to Len(aArray)
       Mostra “Nome “ + aArray[I][1]
       nMedia:=(aArr[I][2]+aArr[I][3]+aArr[I][4]+aArr[I][5])/4
       If nMedia < 10
              cSit := “Reprovado”
       ElseIf nMedia < 25
              cSit := “Exame”
       Else
              cSit := “Aprovado”
       Endif
       Mostra “Resultado: “ + aArr[I][1] + “ “ + cSit
   Next
Return
```

Resposta dos Exercícios

Resposta do Exercicio 3
Usando If… Else…Endif

```
User Function fExFluxo3()
Local aArr:={{"Maria",10,7,15,31},;
             {"Jose",15,16,21,33},;
             {"Petrucio",8,8,8,6},;
             {"Firmino",15,16,21,33},;
             {"Felizberto", 10, 17, 31, 25} }
   For I:=1 to Len(aArray)
       Mostra "Nome " + aArray[I][1]
       nMedia:=(aArr[I][2]+aArr[I][3]+aArr[I][4]+aArr[I][5])/4
       If nMedia < 10
              cSit := "Reprovado"
       ElseIf nMedia < 25
              cSit := "Exame"
       Else
              cSit := "Aprovado"
       Endif
       Mostra "Resultado: " + aArr[I][1] + " " + cSit
   Next
Return
```

Resposta dos Exercícios

Resposta Exercicio 3
Usando Case … EndCase

```
User Function fExFluxo3()
Local aArr:={{"Maria",10,7,15,31},;
             {"Jose",15,16,21,33},;
             {"Petrucio",8,8,8,6},;
             {"Firmino",15,16,21,33},;
             {"Felizberto", 10, 17, 31, 25} }
    For I:=1 to Len(aArray)
        Mostra "Nome " + aArray[I][1]
        nMedia:=(aArr[I][2]+aArr[I][3]+aArr[I][4]+aArr[I][5])/4
        Do Case
           Case nMedia < 10
               cSit := "Reprovado"
           Case nMedia < 25
               cSit := "Exame"
           Otherwise
               cSit := "Aprovado"
        EndCase
        Mostra "Resultado: " + aArr[I][1] + " " + cSit
    Next
```

Resposta dos Exercícios

Resposta do exercicio 4

```
Inicio
        Declarar variável M, I
        Receber Valor da mercadoria em M
        Perguntar se M > 10000
            Se sim, Calcular I := M * 0.15
            Se não, Calcular I := 0
        Imprimir M
        Imprimir I
Fim
```

Resposta dos Exercícios

Resposta do exercicio 5

```
Inicio
        Declarar variável nValSal, nQtdHor, nSalRec
        Declarar variável nMeta := 180
        Receber Valor do Salário em nValSal
        Receber Qtde de Horas em nQtdHor
        Perguntar se nQtdHor > nMeta
           Se sim, Calcular nSalrec := (nValSal+2) * nQtdHor
           Se não, Calcular nSalRec := nValSal * nQtdHor
        Mostrar nSalRec
Fim
```

Resposta dos Exercícios

Resposta do exercicio 6

```
Exercício 6
#include "RWmake.ch"
user function quadrado(aArray)
n1 := aArray[1]^2     // calcula potência de n1
n2 := aArray[2]^2     // calcula potência de n2
n3 := aArray[3]^2     // calcula potência de n3
n4 := aArray[4]^2     // calcula potência de n4
n1 := str(n1)         // converte para string para imprimir no box
n1 += " " + str(n2)   // somar em apenas 1 variável
n1 += " " + str(n4)
    if n3 >= 1000
        msgbox("O Valor do quadrado de n3 :" + chr(13) + chr(10) + str(n3),;
                "Questão 6", "Info")
    else
        msgbox("O Valor do quadro de n1, n2 e n4 são :" + ;
                chr(13) + chr(10) + n1,"Questão 6", "Info")
    endif
```

Resposta dos Exercícios

Resposta do exercicio 7

```
idade := 14
do case
        case idade >= 5 .And.  idade <= 7
                msginfo("Infantil - Classe A")
        case idade >= 8 .And. idade <= 11
                msginfo("Infantil - Classe B")
        case idade = 12 .or. idade = 13
                msginfo("Juvenil - Classe B")
    case idade >= 14 .And. idade <= 17
                msginfo("Juvenil - Classe B")
        case idade < 5
                msginfo("Infantil - Criança só entra acompanhado“)
        otherwise
                msginfo("Adulto")
endcase
```

Configuração do Wizard

Configuração

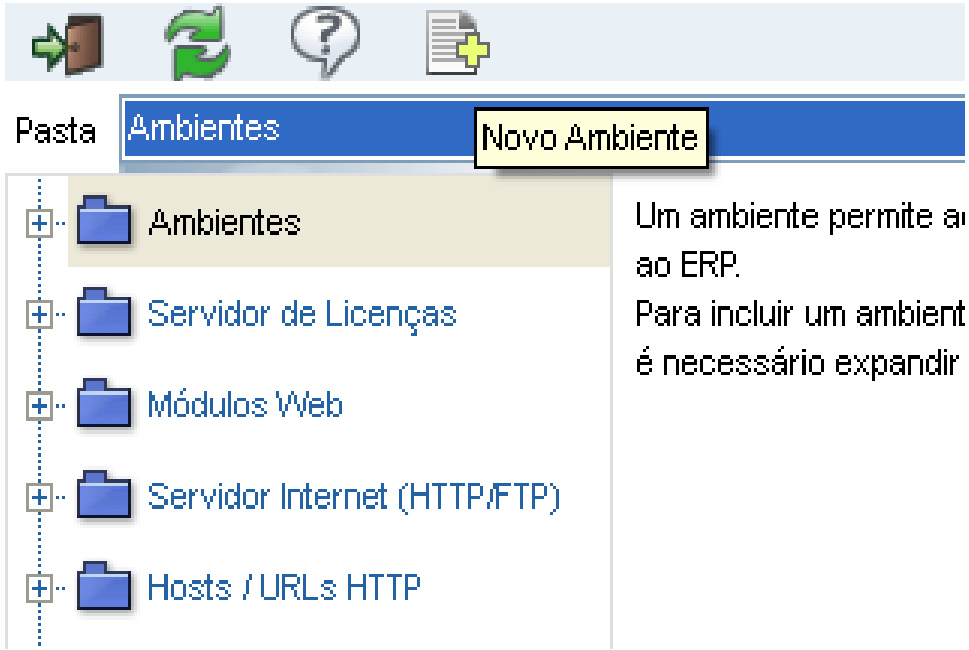
Ambiente: É o onde são realizados os tratamentos das rotinas e suas devidas compilações e que contem os diretórios abaixo.

RootPath: É o diretório onde são armazenados os arquivos do sistema, reconhecida como raiz do ambiente.

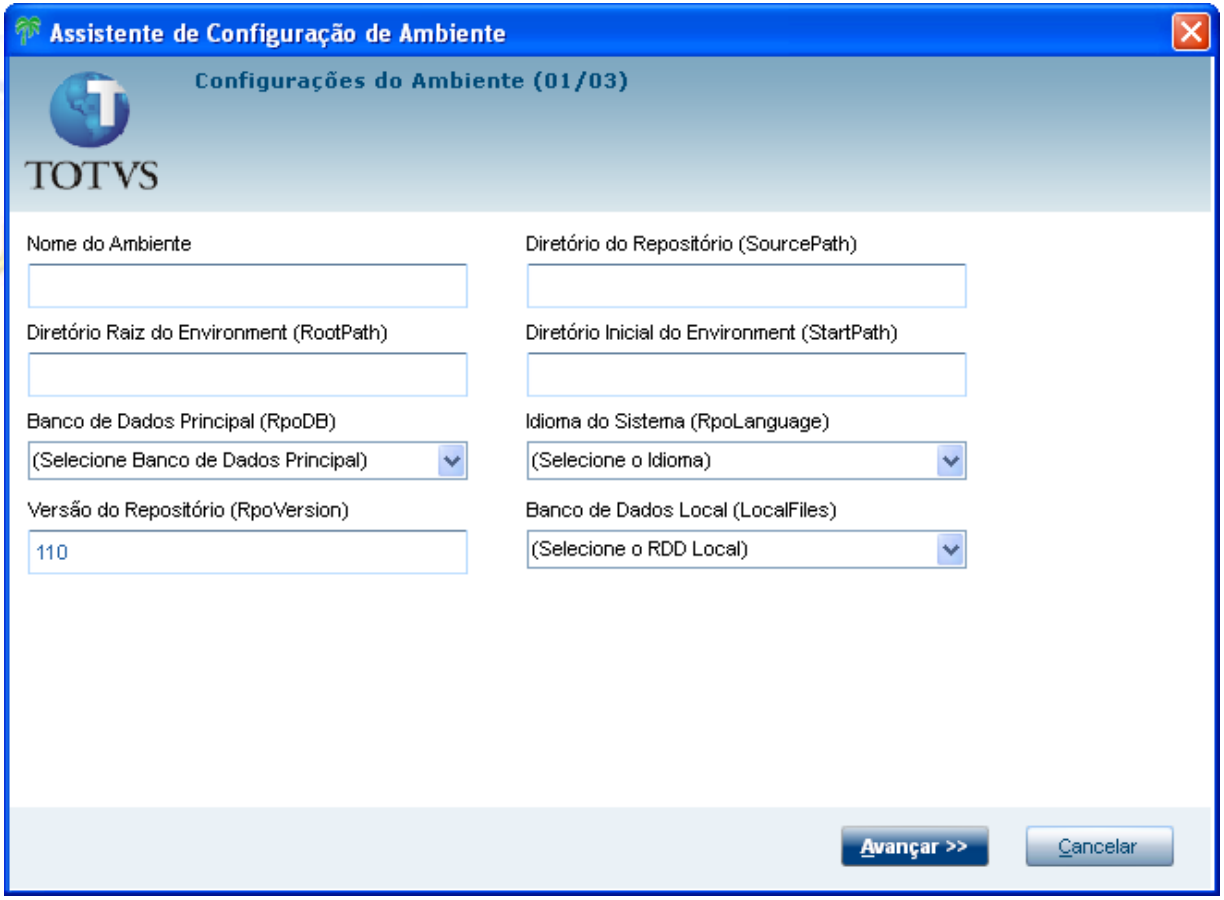
SourcePath: É o diretório onde estão armazenados os repositórios.

StartPath: É o diretório que está dentro do rootpath e que armazena os arquivos de inicialização, banco de dados (.dbf) e menus, senhas de usuarios, cadastro de empresas..

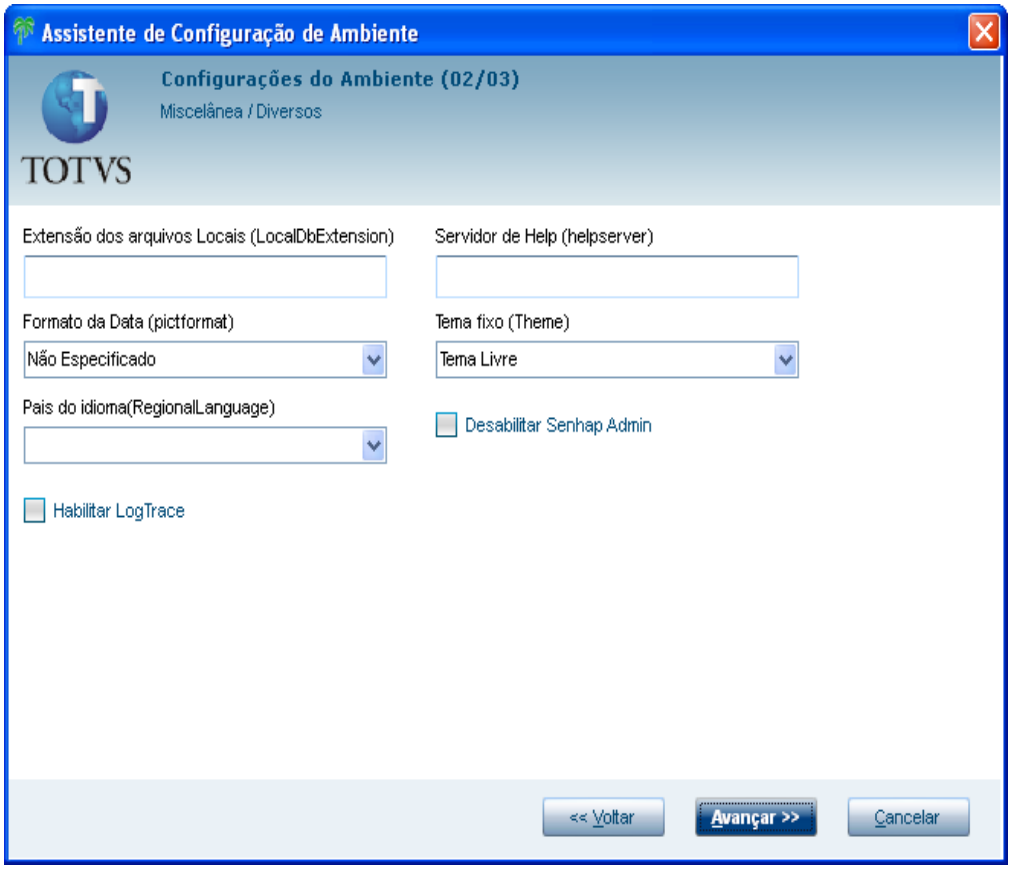
Configuração



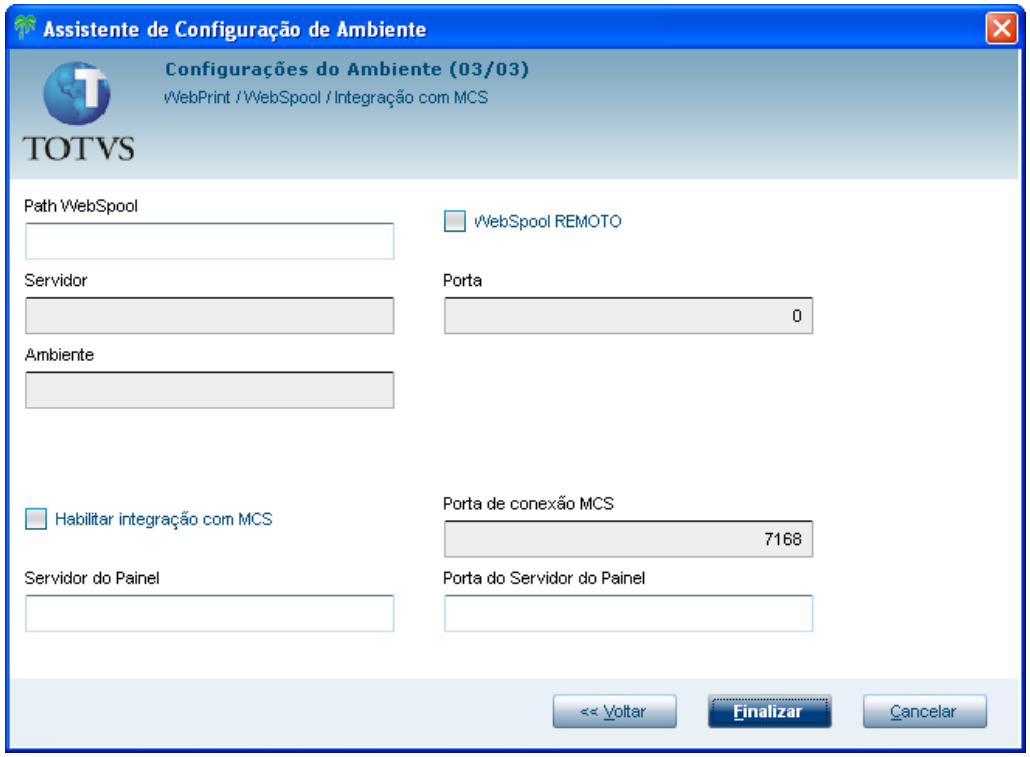
Configuração



Configuração

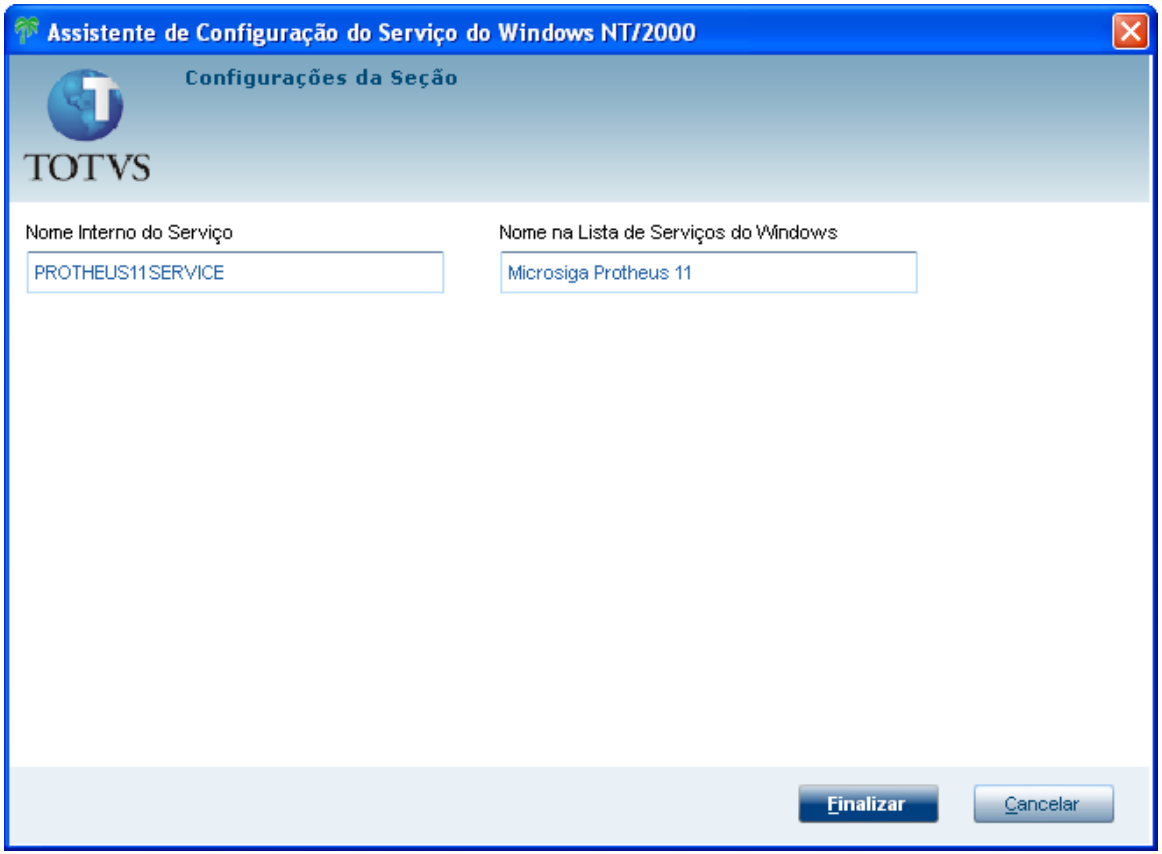


Configuração

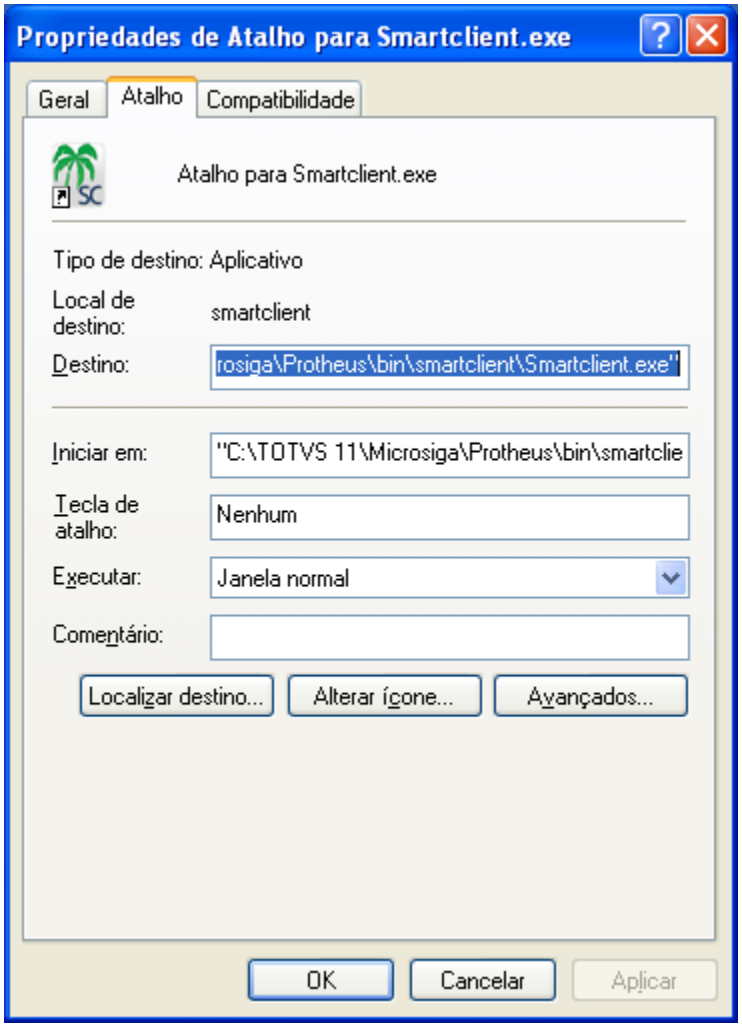


Configuração

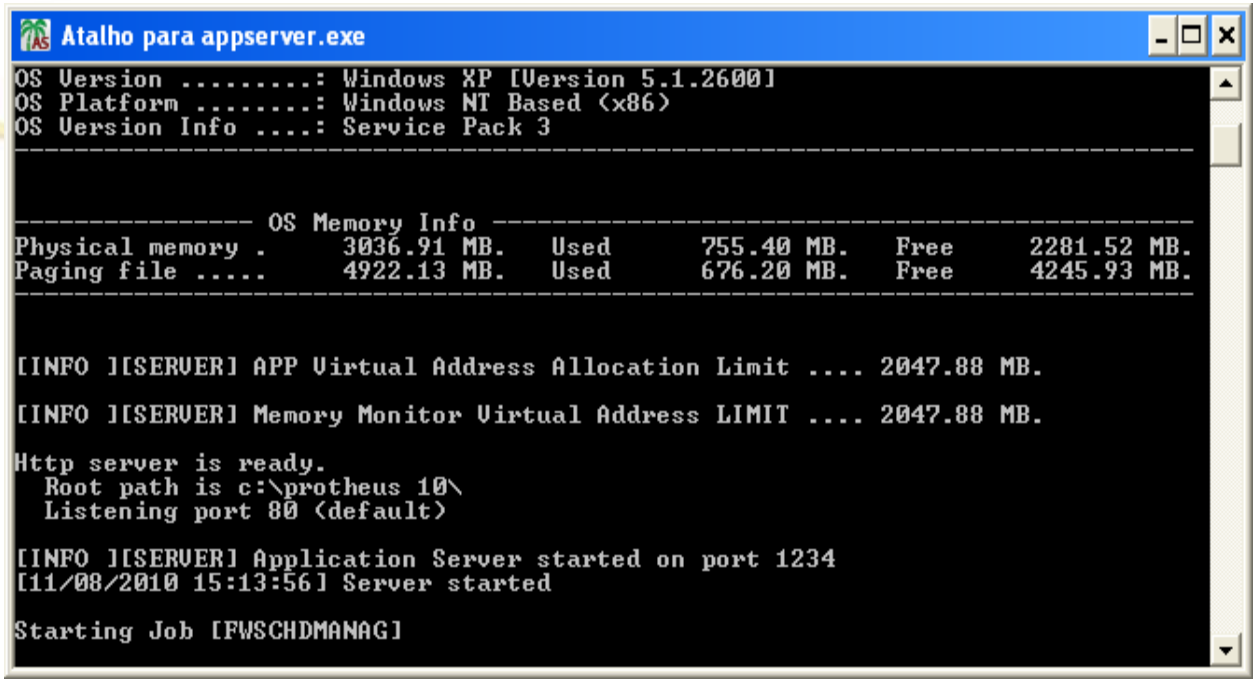
Configuração



Configuração



Configuração



Configurador

Configurador

\PROTHEUS8	Raiz do sistema
\APO	Repositório de objetos (RPO).
\BIN\ACTIVEX	Destinado aos arquivos para acesso via Web
\BIN\REMOTE	Executáveis, bibliotecas e arquivos de configuração (.INI) do sistema.
\BIN\SERVER	Executáveis, bibliotecas e arquivos de configuração (.INI) do sistema.
\BIN\SERVER\ACE_2.50	Arquivos de configuração e bibliotecas para acesso aos arquivos SXs.
\BIN\SERVER\ACE_6.11	Arquivos de configuração e bibliotecas para acesso aos arquivos SXs.
\BIN\SERVER\ACE_6.20	Arquivos de configuração e bibliotecas para acesso aos arquivos SXs.
\BIN\SERVER\ACE_7.00	Arquivos de configuração e bibliotecas para acesso aos arquivos SXs.
\BIN\TOOLS	Onde são encontrados as ferramentas para manutenção do sistema
\CPROVA	Destinado para a gravação dos lançamentos analíticos do módulo Contábil.
\CRYSTAL	Contém arquivos de bibliotecas e relatórios modelos do Crystal Report.
\DATA	Contém o Banco de dados do Protheus (Codebase ou ADS).
\HANDHELD	Arquivos de biblioteca para integração com Palm-OS e Pocket PC
\INCLUDE	Contém as Bibliotecas (.CH) necessárias a execução e compilação do AP7.
\MY PROJECTS\SAMPLES\SOURCE	Fontes para exemplos de funções ADVPL.
\SAMPLES\DOCUMENTS	Arquivos modelos para integração com o pacote Microsoft Office.
\SYSTEMLOAD	Arquivos de carga do Dic. de dados e Helps do Protheus, usado somente na inst.
\SPOOL	Destinado para a gravação de relatórios gerados em disco.
\SEMAFORO	Arquivos complementares para as sequências de documentos.
\SYSTEM	Contém os arquivos de Customização, controle de Empresa e menus do Sistema.
\SISCOMEX	Contém arquivos específicos para uso dos módulos de importação e exportação.
\PROFILE	Armazena o perfil de cada usuário
\SRVWIZARD	Arquivos para execução do Wizard (Configuração do Arquivo MP8SRV.INI)

Configurador

A composição dos nomes das tabelas segue um padrão definido pela microsiga que consiste em:

1° **posição** – (S) de SIGA ou outra letra relacionada com a natureza da tabela.

2° **posição** – de (A à Z) ou de (0 à 9) definindo a família do arquivo.

3° **posição** –- de (1 à Z) definindo a seqüência dentro da família.

“ S “ – Significa que este arquivo pertence aos módulos genéricos.
“ A ” – Letra que representa a família que o arquivo pertence.
“ 1 “ – Seqüência do arquivo na família.
“ 01 “ – Caracteres que representam a numeração da empresa.
“ 0 “ – Dígito de uso exclusivo da Microsiga.

Configurador

SIGAADV.HLP	Help de Campos
SIGAADV.HPR	Help de Programas
SIGAMAT.EMP	Cadastro de Empresas
SIGAMAT.IND	Indice de Empresa
SIGAPSS.SPF	Cadastro de Usuários
SIGAADV.MOT	Motivos de Baixa

Configurador

.#DB	Backup
.REL	Relatórios
.DUP	Script de Duplicata
.PDV	Planilha de Custos
.NF2	Script da Nota Fiscal
.PLN	Planilhas
.RPG	Script do RPG
.DRV	Drivers de Impressoras
.REM	Envio de Transmissão Bancária
.RET	Recebimento de Transmissão Bancária
.FAT	Script de Fatura
.IMP	Configurações de Importação de Dados
.LOG	Arquivo TTS
.PRW e .PRX	Programas em ADVPL e RDMAKE

Dicionários SX

Arquivo	Descrição
SX1	Perguntas e Respostas
SX2	Mapeamento de Tabelas
SX3	Dicionário de Dados
SX5	Tabelas Genéricas
SX6	Parâmetros
SX7	Gatilhos
SX9	Relacionamento entre Tabelas
SXA	Pastas Cadastrais apontadas no SX3
SXB	Consulta Padrão, Utilizada através da tecla F3
SXD	Relatórios disponiveis no schedule de Relatórios do worflow
SXE	Controle de Semáforo – Numero Sequencial (Próximo +1)
SXF	Controle de Semáforo - Numero Sequencial (Próximo)
SXK	Respostas das Perguntas (SX1) por Usuário
SXM	Processos Especiais cadastrados no Schedule do Workflow
SIX	Índices dos Arquivos
.XNU	Menu de Opções dos Módulos

Customização

Parâmetros,
Tabelas,
Perguntas,
Fórmulas - Expressões em AdvPL / User Function,
Lançtos Padrões / Expressões AdvPL/User Function,
Validações,
Gatilhos,
Campos de Arquivos,
User Function via menu,
Pontos de Entrada,
Dicionário de Dados Ativo,
SigaRPM,
Crystal,
Integração Office (Word , Excel)

Customização

Parâmetros

GETMV - Retorna o conteúdo de um parâmetro cadastrado no SX6.
Sintaxe
GETMV(*cNomPar,* [*lRetPar*], [*uRet*])

PUTMV – Grava informação no parâmetro no SX6.
Sintaxe
PUTMV(*cNomPar, cConteudo*)

Customização

Criar Parâmetro pelo Configurador

Parametro: AD_DATAFIN
Descricao: Data Inicio de operacoes
Conteúdo: 01/01/2007

Rodar em fórmulas:

```
Msginfo(GETMV("AD_DATAFIN"))
putmv("AD_DATAFIN",ctod("01/05/2007"
))
Msginfo(GETMV("AD_DATAFIN"))
```

Customização

Criar Parâmetro pelo Configurador

Parametro: AD_DATAFIN
Descricao: Data Inicio de operacoes
Conteúdo: 01/01/2007

Rodar em fórmulas:

```
Msginfo(GETMV("AD_DATAFIN"))
putmv("AD_DATAFIN",ctod("01/05/2007"))
Msginfo(GETMV("AD_DATAFIN"))
```

Customização

Criar tabela pelo Configurador

Tabela: Z1
Descrição: Dias da semana
Chave: 1 – Domingo
Chave: 2 – Segunda
Chave: 3 – Terça
Chave: 4 – Quarta
Chave: 5 – Quinta
Chave: 6 – Sexta
Chave: 7 - Sábado

Rodar em fórmulas:

```
Msginfo(TABELA( “Z1”,”2” ) )
msginfo(TABELA("Z1",cValToChar(DOW(dDATABASE))))
```

Customização

Perguntas

PERGUNTE – Carrega e Monta a tela de perguntas cadastradas no SX1.

Sintaxe

PERGUNTE(*cGrupo, lMostra*)

Customização

Gatilho

Criar um gatilho no configurador
Clientes
Ao digitar o Nome deve preencher o
Nome Reduzido com o nome

Testar no cadastro de clientes, se não
tiver o cadastro de clientes
No menu, incluir a funcao MATA030

Customização

Criar um arquivo pelo configurador

SZ5 – Cadastro de Alunos
Z5_FILIAL C 2
Z5_CODIGO C 4 – Usado – Browse – Obrig.
Z5_NOME C 30 – Usado – Browse – Obrig.
Z5_ENDERECO C 30 – Usado -
Z5_APELIDO C 20 - Usado
Z5_TELEFONE C 15 - Usado
Z5_DATANASC D 8 – Usado - Browse

Indice 1 – filial + Codigo
Indice 2 – Filial + nome

Customização

✓ **Consulta Padrão**

Criar uma Consulta Padrão do Cadastro de Produtos

Nome Consulta	Tabela	Retorno	Filtro
SB1NEW	SB1	SB1->B1_COD	SB1->B1_TIPO == “PA”

Índice	Colunas
Filial + Código	Código, Descrição
Filial + Grupo	Grupo, Código, Descrição
Filial + Cod. Barras	Cod. Barras, Código, Descrição

Alterar Consulta Padrão do campo C6_PRODUTO para SB1NEW e testar com o programa MATA460() em formulas.

Customização

Criação de Menu

Criar um menu idêntico ao menu faturamento
Dar seu nome ao menu.

Customização

Criação de Grupo

Criar um grupo chamado TI
Com acesso a todas as empresas e acesso aos módulos faturamento e compras

Criação de Usuário

Criar um usuário com seu nome
Adicionar o grupo TI.
Coloque o menu que você criou no menu de faturamento.

IDE

Integrated Development Environment

IDE

Configuração de Ambiente – Podemos configurar o ambiente que será atualizado na compilação e depuração de programas.

IDE

Criação do Projetos – Criamos projetos como forma de organizar os programas criados de modo que qualquer outro analista possa entender o roteiro das customizações;

Criação do programa – Através de edição podemos criar programas;

Compilação – Compilamos os programas ou todo o projeto para enviarmos ao repositório o que foi mudado e para que o protheus saiba o que queremos que faça;

Depuração – Execução passo a passo das rotinas, permitindo que o analista veja o andamento do programa e o conteúdo das variáveis, bem como as tabelas abertas;

IDE

Utilizando o IDE

Assistente de Código

Criando o primeiro Cadastro

- Utilizar a tabela criada SZ5
- Incluir no projeto
- Compilar
- Incluir o seu programa no menu com seu nome.

IDE

Assistente de Código

Criando o primeiro Cadastro

- Utilizar a tabela criada SZ5
- Incluir no projeto
- Compilar
- Incluir o seu programa no menu com seu nome.

Manipulando Mensagens

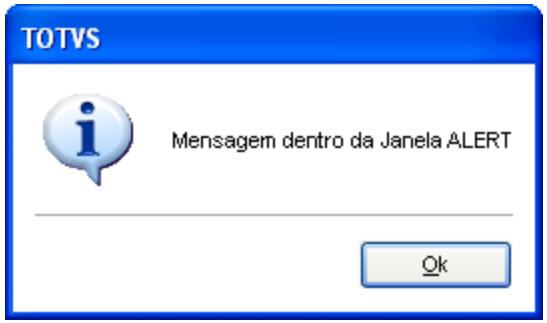
MSGBOX – Mostra mensagem na tela.
Sintaxe:
MSGBOX(cMensagem,cTítulo,cTpCaixa)

Argumento	Obrigat.	Tipo	Descrição
cMensagem	Sim	C	Mensagem dentro da janela
cTitulo	Não	C	Titulo da Janela
cTpCaixa	Não	C	Tipo da Caixa

Tipos de caixas:
“STOP”, utiliza um bitmap para advertência e tem um botão “Ok”. Retorna Nil.
“INFO”, utiliza um bitmap para advertência e tem um botão “Ok”. Retorna Nil.
“ALERT”, utiliza um bitmap para advertência e tem um botão “Ok”. Retorna Nil.
“YESNO”, utiliza um bitmap para advertência e tem dois botões “Sim” e “Não”, retorna .T. ou .F.
“RETRYCANCEL”, utiliza um bitmap para advertência e tem dois botões “Repetir” e “Cancelar”, retorna .T. ou .F.

Manipulando Mensagens

Exemplo:
MSGBOX(“Mensagem dentro da Janela ALERT”, “Titulo da Janela”, “ALERT”)



Manipulando Mensagens

AVISO – Mostra mensagem na tela com opção de escolha.

Sintaxe:

AVISO(cTitJan,cCorpo, ,aOpcoes, ntp, cTitMens)

Argumento	Obrigat.	Tipo	Descrição
cTitJan	Sim	C	Titulo da janela
cCorpo	Sim	C	Corpo da Mensagem
aOpcoes	Sim	A	Array com Botões da Janela
nTp	Sim	N	1 – Pequena, 2- Média, 3- Grande
cTitMens	Não	C	Titulo da Mensagem

Manipulando Mensagens

Exemplo:

```
nOpc := AVISO("Titulo da Janela", "Corpo
da Mensagem", ;
                {"Sim", "Nao", "Talvez"},1,
"Titulo da Mensagem")
If nOpc == 1
        msginfo("voce escolheu sim",
"Titulo")
ElseIf    nOpc == 2
        msginfo("voce escolheu nao",
"Titulo")
Else
        msginfo("voce escolheu talvez",
"titulo")
Endif
```

Manipulando Mensagens

Outras funções de mensagem:

- Msginfo()
- Alert()
- Msgalert()
- Msgrun()
- Msaguarde()

Manipulação de String

VAL - Converte texto em valor.
Sintaxe:
VAL(cValor)

Argumento	Obrigat.	Tipo	Descrição
cValor	Sim	C	Conteúdo que se quer converter.

Exemplo:
cCodigo := “000015”
nNum := Val(cCodigo)+1
msginfo(nNum) // nNum terá o valor de 16

Manipulação de String

LEFT – Retorna o conteúdo de uma qtde de caracteres a esquerda
Sintaxe
LEFT(cConteudo, nCount)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer extrair uma parte
nCount	Sim	N	Qtde de caracteres a extrair

Exemplo:
cCodigo := “Paralelepipedo”
cPedaco1 := Left(cCodigo,5) // cPedaco1 terá o conteúdo “paral”

Manipulação de String

RIGHT – Retorna o conteúdo de uma qtde de caracteres a direita
Sintaxe
RIGHT(cConteudo, nCount)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer extrair uma parte
nCount	Sim	N	Qtde de caracteres a extrair

Exemplo:
cCodigo := “Paralelepipedo”
cPedaco1 := Right(cCodigo,5) // cPedaco1 terá o conteúdo “ipedo

Manipulação de String

PADC – Centraliza um texto conforme a qtde de caracteres especificados.
Sintaxe
PADC(cConteudo, nCount)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer centralizar
nCount	Sim	N	Qtde de caracteres a centralizar

Exemplo:
cCodigo := “Parede”
cCodigo := Padc(cCodigo,14)
// cCodigo terá o conteúdo “ Parede ”

Manipulação de String

PADR – Preenche com espaços em branco a direita conforme a qtde de caracteres especificados.

Sintaxe

PADR(cConteudo, nCount)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer alinhar
nCount	Sim	N	Qtde de caracteres a alinhar

Exemplo:

```
cCodigo   := “Parede”
cCodigo  := Padr(cCodigo,14)
// cCodigo  terá o conteúdo “       Parede”
```

Manipulação de String

ALLTRIM – Limpa espaços em branco iniciais e finais

Sintaxe

Alltrim(cConteudo)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer limpar

Exemplo:

cCodigo := “ Parede ”

cCodigo := Alltrim(cCodigo) // cCodigo terá o conteúdo “Parede

Manipulação de String

LTRIM – Limpa espaços em branco a esquerda
Sintaxe
Ltrim(cConteudo)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer limpar

Exemplo:
cCodigo := “ Parede ”
cCodigo := Ltrim(cCodigo) // cCodigo terá o conteúdo “Parede ”

Manipulação de String

RTRIM – Limpa espaços em branco a direita

Sintaxe

Rtrim(cConteudo)

Argumento	Obrigat.	Tipo	Descrição
cConteudo	Sim	C	Conteúdo que se quer limpar

Exemplo:

cCodigo := “ Parede ”

cCodigo := Ltrim(cCodigo) // cCodigo terá o conteúdo “ Parede”

Manipulação de String

STR – Converte numero em caracter
Sintaxe
Str(nConteudo)

Argumento	Obrigat.	Tipo	Descrição
nConteudo	Sim	N	Valor a converter

Exemplo:
nCodigo := 1
cCodigo := Str(nCodigo) // cCodigo terá o conteúdo “ 1”

Manipulação de String

STRZERO – Converte numero em caracter, preenchendo com zeros a esquerda.
Sintaxe
Strzero(nConteudo, nCount)

Argumento	Obrigat.	Tipo	Descrição
nConteudo	Sim	N	Valor a converter
nCount	Sim	N	Qtde de caracteres

Exemplo:

```
nCodigo   := 1
cCodigo  := Strzero(nCodigo, 6)  // cCodigo  terá o conteúdo “000001”
nCodigo   := 158
cCodigo  := Strzero(nCodigo, 6)  // cCodigo  terá o conteúdo “000158”
```

Manipulação de String

TRANSFORM – Converte numero em caracter utilizando máscara.
Sintaxe
Transform(nConteudo, cMask)

Argumento	Obrigat.	Tipo	Descrição
nConteudo	Sim	N	Valor a converte
CMask	Sim	N	Formato do caracter

Exemplo:

```
nCodigo   := 1500
cCodigo  := “Valor: ” + Transform(nCodigo, “@E 99,999.99”)
// cCodigo  terá o conteúdo (Valor:   1.500,00)
```

Manipulação de Data

DATE – Retorna a data do sistema operacional
Sintaxe
DATE()

Exemplo:
dDataAtual := date()
// dDataAtual terá o conteúdo da data de hoje

Manipulação de Data

DTOS – Converte data em String no formato AAAAMMDD
Sintaxe
DTOS(dData)

Argumento	Obrigat.	Tipo	Descrição
dData	Sim	D	Data a converter

Exemplo:
cData := dtos(date()) // cData terá o conteúdo 20061231

Manipulação de Data

DTOC – Converte data em Caracter no formato DD/MM/AA.

Sintaxe

DTOC(dData)

Argumento	Obrigat.	Tipo	Descrição
dData	Sim	D	Data a converter

Exemplo:

cData := dtoc(date()) // cData terá o conteúdo 31/12/06

Manipulação de Data

CTOD – Converte Caracter no formato DD/MM/AA em Data.

Sintaxe

CTOD(cData)

Argumento	Obrigat.	Tipo	Descrição
cData	Sim	C	Data a converter

Exemplo:

cData := “31/12/06”

cData := ctod(cData) // cData terá o conteúdo 31/12/06

Manipulação de Data

STOD – Converte string no formato AAAAMMDD em Data.

Sintaxe

STOD(cData)

Argumento	Obrigat.	Tipo	Descrição
cData	Sim	C	Data a converter

Exemplo:

cData := “20061231”

cData := stod(cData) // cData terá o conteúdo 31/12/06

Manipulação de Data

MONTH – Retorna o Mes
Sintaxe
Month(dData)

Argumento	Obrigat.	Tipo	Descrição
dData	Sim	D	Data

Exemplo:
nMes := Month(date())
// nMes terá o conteúdo 12 para a data 31/12/06

Manipulação de Data

DAY – Retorna o dia

Sintaxe

Day(dData)

Argumento	Obrigat.	Tipo	Descrição
dData	Sim	D	Data

Exemplo:

```
nDia  := Day(date())
// nDay  terá o conteúdo 31 para a data 31/12/06
```

Manipulação de Data

YEAR – Retorna o ano
Sintaxe
Year(dData)

Argumento	Obrigat.	Tipo	Descrição
dData	Sim	D	Data

Exemplo:
nAno := Year(date())
// nAno terá o conteúdo 2006 para a data 31/12/06

Manipulação de Data

MESEXTENSO– Retorna o Mês por extenso em português

Sintaxe

MesExtenso(dData)

Exemplo:

```
cMes  := MesExtenso(date())
// cMes  terá o conteúdo Dezembro para a data 31/12/06
```

IDE

Assistente de Código

Criando o primeiro relatório

Assistente do código

Cadastro de produtos (ordem 1)

Campos:

Código (B1_COD)

Descrição (B1_DESC)

Tipo (B1_TIPO)

Inserir no menu com seu nome pelo configurador

Rodar o relatório

Blocos de código

Várias instruções em uma única linha de programação.

Exemplo:
N1 := 6
N2 := 20
N3 := N1 * N2

Em Bloco de Código

bVar := {|| N1 := 6, N2 := 20, N3 := N1 * N2}

Para Executar um bloco usa-se a função Eval()

nVar := Eval(bVar) -> o retorno é sempre o resultado da ultima

expressão do bloco

Blocos de código

Passagem de parâmetros para o Bloco

Os parâmetros são passados em um bloco através do módulo (||), similar a uma função, separando cada argumento por virgula.

Exemplo:

```
nVar1 := 5
nVar2 := 8
bVar  := {|X,Y|  nTudo := X * Y}

nValor := Eval(bVar,nVar1, nVar2)
```

Equivalente a:

```
nTudo := 5 * 8
```

nValor será igual a 40

Blocos de código

Funções:

Eval(bVar,Par1,Par2...Parx)

Executa um bloco de código qualquer, com passagem de parâmetros.

dbEval(bVar,bFor,bWhile)

Executa um bloco de código enquanto obedecer a condição.

aEval(aArray,bBloco,nInicio,nCount)

Executa um bloco de código, passando um array como parâmetro, a
função percorre o array de acordo com os argumentos de inicio e
quantidade de vezes do array.

Blocos de código

SA - Cadastros
SB - Estoques
SC - Previsões de E/S
SD - Mov. De Estoque
SE - Financeiro
SF - Fiscal
SG - Estruturas
SH - Carga de Máquina
SI - Contábil
SJ - Estatísticas
SM - Miscelâneas
SN - Ativo Fixo
SO - Assist. Técnica
SP - Ponto Eletrônico
SQ - Recr. e Seleção
SR - Folha de Pagto
ST - Manut. Industrial
SU - Telemarketing
SV - Concessionárias
SW e SY - Export./Import.
Q? - Qualidade (Celerina)
R? - Recursos Humanos
SZ,QZ,RZ - Livres
P? - Projetos Especiais

Nomenclatura

O sistema permite controlar até 99 Empresas. Cada Empresa pode ter até 99 Filiais:

Empresa 01 – Filiais 01
02
...
99

Empresa 02 – Filiais 01
02
...
99

Para cada Empresa é criado um jogo de arquivos:

SXXnn0 – onde: XX = Prefixo do arquivo
nn = Empresa

Exemplo: Empresa 99 -> SA1990, SA2990, SB1990...

Conceitos filiais

Filiais

Os dados das Filiais ficam dentro do arquivo de cada Empresa.

Todo arquivo, exceto SM2, tem um campo XX_Filial que identifica a Filial: 01,01,01,02,02,02

Nomenclatura dos campos:

Arquivo:	SA1	SA2	SB1	QA1
	A1_FILIAL	A2_FILIAL	B1_FILIAL	QA1_FILIAL
	A1_COD	A2_COD	B1_COD	QA1_COD
	A1_NOME	A2_NOME	B1_DESC	QA1_DESC
	A1_END	A2_END	B1_TIPO	

Acesso a Filiais

Compartilhado → registros compartilhados entre as filiais
Campo Filial: em branco

Exclusivo → o registro é exclusivo da filial
Campo Filial: código da Filial (variável cFilAnt)

Ex: SA1 – Compartilhado

A1_FILIAL A1_COD A1_NOME

000015 FABRICA DE TINTAS E VERINIZES LTDA /SP
000016 TEX MALHAS E CONFECCOES S.A /SP
000001 CLIENTE PADRAO
000020 ARMANDO JOSE FLORES /SC
000007 COMERCIO SOM DO MUNDO /SP

xFilial("SA1") → " "

Ex: SC5 - Exclusivo

C5_FILIAL	C5_NUM	C5_EMISSAO	C5_CLIENTE	C5_PROD	C5_VALOR
01	000001	01/01/04	000002	11.11014	15.000,00
01	000005	01/01/04	000023	13.13000	5.600,00
01	000007	20/01/04	000016	15.15140	5.600,00
02	000001	01/01/04	000002	11.11014	15.000,00
02	000005	01/01/04	000023	13.13000	5.600,00
02	000007	20/01/04	000016	15.15140	5.600,00
03	000001	01/01/04	000002	11.11014	15.000,00
03	000005	01/01/04	000023	13.13000	5.600,00
03	000007	20/01/04	000016	15.15140	5.600,00

xFilial("SC5") → cFilAnt

Funções de Base de dados

CRIATRAB - Cria arquivo de trabalho, com nome temporário.
Sintaxe: CriaTrab([aCampos], lExclus)

Argumento	Obrigat.	Tipo	Descrição
aCampos	Nao	A	É o array com os campos da tabela a criar. Quando nulo apenas retorna um nome temporário e não cria a tabela.
lExclus	Sim	L	.T. cria tabela exclusiva .F. cria tabela compartilhada

Funções de Base de Dados

Exemplo:

```
Local aCampos := {{'T_COD' ,"C",6, 0},;
                  {'T_NOME' ,"C",30, 0},;
                  {'T_ENDERECO' ,"C",30,0},;
                  {'T_DATA' ,"D",8,0},;
                  {'T_NUMERO' ,"N",17, 2}}

//Cria o arquivo fisicamente em DBF no RothPath
cArqTra := CriaTrab( aCampos, .T. )
dbUseArea(.t.,,cArqTra,"TMP",.f.,.f.)

//Somente gera um nome temporario
cArqTemp := CriaTrab( Nil, .F. )
```

Índice

Um arquivo pode ter até 35 índices (1...9, A...Z)
dbSetOrder(n), onde n = 1...35 → função para selecionar o índice.
Além dos índices originais, usuários podem incluir seus próprios índices.
Caso a Microsiga inclua novos índices, os do usuário serão renumerados

SA1 – índices do Sistema	1	
	2	
	...	
	7	
índice do Usuário	8	A1_TIPO

Nos programas: dbSetOrder(8)

Portanto, índices de usuário devem sempre ter um NickName. Deve-se usar a função dbOrderNickName("NICKNAME")
Índices do sistema, nunca terão NickName. Usa-se dbSetOrder().

Funções de Banco de Dados

INDREGUA - Cria índice temporário.

Sintaxe: INDREGUA(cAlias, cArqNtx, cIndCond,[cTipo],;
[cFiltro], cMensagem, [lMostra])

Argumento	Obrigat.	Tipo	Descrição
CAlias	Sim	C	Alias do arquivo que quer criar índice
cArqNtx	Sim	C	Nome físico do índice
cIndCond	Sim	C	Chave do índice
CTipo	Nao	C	Tipo de índice “C” Crescente ou “D” Decrescente, se não informado o Default é “C” crescente.
cFiltro	Não	C	Filtro desejado
cMensagem	Sim	C	Mensagem a aparecer na tela
lMostra	Não	L	.T. Mostra Régua de progressão (default) .F. Não mostra a régua

Funções de Banco de Dados

Exemplo:

```
//Somente gera um nome temporario
cArqTemp := CriaTrab( Nil, .F. )
cChave   := "B1_FILIAL+B1_GRUPO"
cCpo     := "B1_COD"
cMens    := "Selec.registros..."
IndRegua("SB1",cArqTemp,cChave,,cCpo,"D",cMens, .T.)

//Retorna informações originais
SB1->(DbSetOrder(1)) //Posicionando na ordem de origem

//Deletando arquivo de trabalho
fErase(cArqtemp+OrdBagExt())

//Restaura indices padrões
RetIndex("SB1")
```

Funções de Banco de Dados

DBUSEAREA – Abre uma tabela e a deixa corrente.
Sintaxe: DBUSEAREA(lNew, cDriver, cNomArq, cAlias, lShared,lReadOnly)

Argumento	Obrigat.	Tipo	Descrição
lNew	Não	L	.T. Abre em uma nova Area .F. (Default) Abre na área corrente, fechando a área anteriormente utilizada.
cDriver	Não	C	Driver da tabela a abrir. Default DBFCDX
cNomArq	Sim	C	Nome da tabela a abrir
cAlias	Sim	C	Apelido
lShared	Não	L	.T. Abre a tabela compartilhada .F. (Default) Abre a tabela exclusiva
lReadOnly	Não	L	.T. Abre a tabela somente leitura .F. (Default) Permite gravação na tabela

Funções de Banco de Dados

DBSELECTAREA – Seleciona uma Área de Trabalho.
Sintaxe: DBSELECTAREA(cAlias)

Argumento	Obrigat.	Tipo	Descrição
cAlias	Sim	C ou N	Se numérico, seleciona área pelo número, se Caracter, seleciona a área pelo Apelido.

DBSETORDER – Seleciona um índice pela ordem.
Sintaxe: DBSETORDER(nOrd)

Argumento	Obrigat.	Tipo	Descrição
nOrd	Sim	N	Número do Indice

Funções de Banco de Dados

DBSKIP – Salta registro da tabela corrente.
Sintaxe: DBSKIP(nStep)

Argumento	Obrigat.	Tipo	Descrição
nStep	Nao	N	Default 1. Qtde de registros que será saltado. Se positivo avança registros, Se negativo retrocede registros.

DBGOTO – Posiciona no registro especificado da tabela corrente.
Sintaxe: DBGOTO(nReg)

Argumento	Obrigat.	Tipo	Descrição
bFiltro	Sim	B	Bloco com a expressão de filtro
cFiltro	Sim	C	Expressão do Filtro

Funções de Banco de Dados

DBSETFILTER – Retorna a expressão do filtro da tabela corrente.
Sintaxe: DBSETFILTER(bFiltro, cFiltro)

Argumento	Obrigat.	Tipo	Descrição
bFiltro	Sim	B	Bloco com a expressão de filtro
cFiltro	Sim	C	Expressão do Filtro

Funções de Banco de Dados

DBGOTOP – Posiciona o ponteiro no primeiro registro de acordo com a ordem corrente.
Sintaxe: DBGOTOP()

DBGOBOTTOM – Posiciona o ponteiro no ultimo registro de acordo com a ordem corrente.
Sintaxe: DBGOBOTTOM()

DBFILTER – Retorna a expressão do filtro da tabela corrente.
Sintaxe: DBFILTER()

DBCLEARFIL – Limpa o filtro da tabela corrente.
Sintaxe: DBCLEARFIL()

Funções de Banco de Dados

```
dbUseArea(.T., “TOPCONN”, “SX5020”, “SX5A”, .T., .T.)
dbSelectArea(“SB1”)
dbSetorder(1)
dbGotop()
cFiltro := “B1_TIPO > ‘MP’ ”
cADFilAnt := SB1->(dbFilter())
SB1->(dbSetFilter({||&cFiltro},cFiltro))
While SB1->(!EOF())
        If !dbSeek(xFilial(“SB1”) + SB1->B1_COD)
                        SB5->(dbAppend())
        Endif
        SB5->B5_FILIAL := xFilial(“SB1”)
        SB5->B5_COD := SB1->B1_COD
        SB1->(MsUnlock())
    SB1->(dbSkip())
Enddo
If Empty(cADFilAnt)
        SB1->(dbClearFil())
Else
        SB1->(dbSetFilter({||&cADFilant},cADFilant))
Endif
```

Funções de Banco de Dados

Função Posicione

Sintaxe:
Posicione(cAlias, nOrdem, cChave, cCampo)

Exemplo:
Posicione("SB1", 1, xFilial("SB1") + cCodigo, "B1_DESC")

Funções de Banco de Dados

Função Existcpo

Retorna se determinada chave existe ou não no arquivo.
Se existir retorna .T.
Se não existir retorna .F.

Sintaxe :
ExistCpo(cAlias,cChave,nOrdem)

Exemplo :
ExistCpo("SB1", cCodigo, 1)

Funções de Banco de Dados

Função Existchav
Retorna se determinada chave existe ou não no arquivo.
Se existir retorna .F.
Se não existir retorna .T.

Sintaxe :
ExistChav(cAlias,cChave,nOrdem)

Exemplo :
ExistChav("SB1", cCodigo, 1)

IDE

ATIVOS	ATIVOS DIVERSOS	GG
TOTAL DE GG		1
LUVA	LUVA PARA PROTECAO DAS MAOS	MC
MANUTENCAO	ITEM PARA CONTROLE DE MANUTENC	MC
TERCEIROS	ITEM PARA CONTROLE DE MANUTENC	MC
TOTAL DE MC		3
MOD0001	MAO DE OBRA FABRICACAO	MO
MOD0002	MAO DE OBRA DO ACABAMENTO	MO
TOTAL DE MO		2
ARGOLA	ARGOLA PARA PRENDER CHAVES	MP
ESCUDO	ESCUDO COM LOGOTIPO SIGA	MP
MOSQUETAO	PRENDEDOR DO CHAVEIRO	MP
REBITE	REBITE DE FERRO	MP
SUPORTE	SUPORTE DE COURO	MP
TIRANTE	TIRANTE DE COURO	MP
TOTAL DE MP		6
CHAVEIRO	CHAVEIRO DE BRINDE MICROSIGA	PA
TOTAL DE PA		1
CORPO	CORPO DE COURO	PI
TOTAL DE PI		1
TOTAL GERAL:		

DICAS

É sempre bom utilizar o Alias antes de
comando de Base de Dados.

SB1->(DBGOTOP()) – Posiciona no primeiro registro do arquivo
SB1 de acordo com a ordem que esta selecionada no momento.

SB1->(DBGOBOTTON()) – Posiciona no ultimo registro do arquivo
SB1 de acordo com a ordem que esta selecionada no momento.

SB1->(DBSEEK(XFILIAL() + “000100”)) - Busca em SB1 o registro
que obedeça a chave estipulada de acordo com a ordem
selecionada no momento.

Ao executar um DBSEEK(), verificar se localizou o registro.
Exemplo:

```
If ! SB1->(dbSeek(xFilial("SB1")))
   // Não achei o registro
Endif
```

DICAS

Quanto ao comando DO WHILE não esquecer de incluir a condição referente à filial, quando esta leitura for de registros de uma filial).

Exemplo :

```
dbSelectArea("SB1")
SB1->(dbSeek(xFilial("SB1")))
Do While SB1->(! Eof() .And. B1_FILIAL == xFilial("SB1"))
        // Processamento
        SB1->(dbSkip())
Enddo
```

Ao criar uma função que irá desposicionar registros, use a função GETAREA() e RESTAREA(), para voltar tudo à posição original.

Exemplo:

```
Dbselectarea("SD1")
aAreasd1 := Getarea() // Armazena o ambiente do arquivo SD1
SD1->(dbsetorder(3))
SD1->(dbseek(xfilial("SD1") + DTOS("01/03/01"), .T.))
Do While ! Eof() .And. D1_FILIAL == xfilial("SD1") ;
         .And. D1_EMISSAO <= mv_par02
       // Processamento
       Dbskip()
Enddo
Restarea(aAreasd1) // Restaura o ambiente do arquivo SD1
```

Pontos de Entrada

São aberturas nas rotinas padrões do sistema que deixa o sistema flexível permitindo o desenvolvimento de processos específicos a partir de uma rotina padrão do sistema.

Permite maior abrangência nos diversos segmentos, atendendo as necessidades do cliente e dos analistas de campo.

Pontos de Entrada

São aberturas nas rotinas padrões do sistema que deixa o sistema flexível permitindo o desenvolvimento de processos específicos a partir de uma rotina padrão do sistema.

Permite maior abrangência nos diversos segmentos, atendendo as necessidades do cliente e dos analistas de campo.

Pontos de Entrada

Exemplo:
Criação de módulo Especifico: ESPNOME

User Function ESPNOME()

Return “Meu Módulo”

Pontos de Entrada

Exemplo:
Após a Inclusão do Produto: MT010INC

User Function MT010INC()

```
msginfo("Você acabou de incluir um produto!")
```

Return()

Pontos de Entrada

Antes da exclusão do Produto: MTA010OK
Retorno .T. – Permite excluir
Retorno .F. – Não Permite Excluir

```
User Function MTA010OK()
Local lRet := .T.
   If dDataBase <  ctod("01/01/2010")
            msginfo("Voce não pode excluir esse produto")
            lRet := .F.
   Endif
Return lRet
```

Exercícios

11 - Crie um cadastro de funcionários SZ6, com os campos do teste de mesa pag.18.

Crie um rdmake para cadastrar os funcionários
Crie um rdmake para mostrar o salário reajustado na tela.

Outras funções

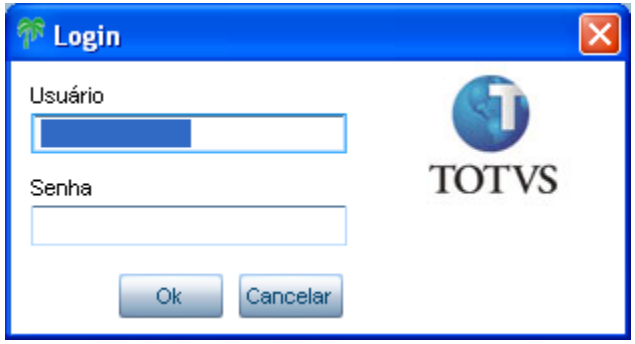
Usando Norton Guide

Funcoes.ng -> Clipper em Português

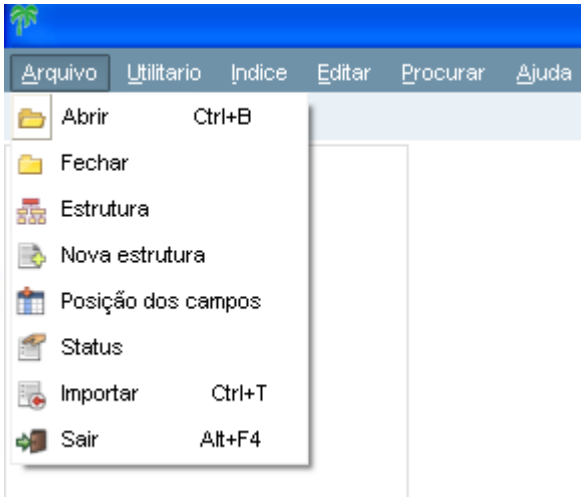
C53G01C.ng -> Clipper em Inglês

APSDU OU MPSDU

Outras funções

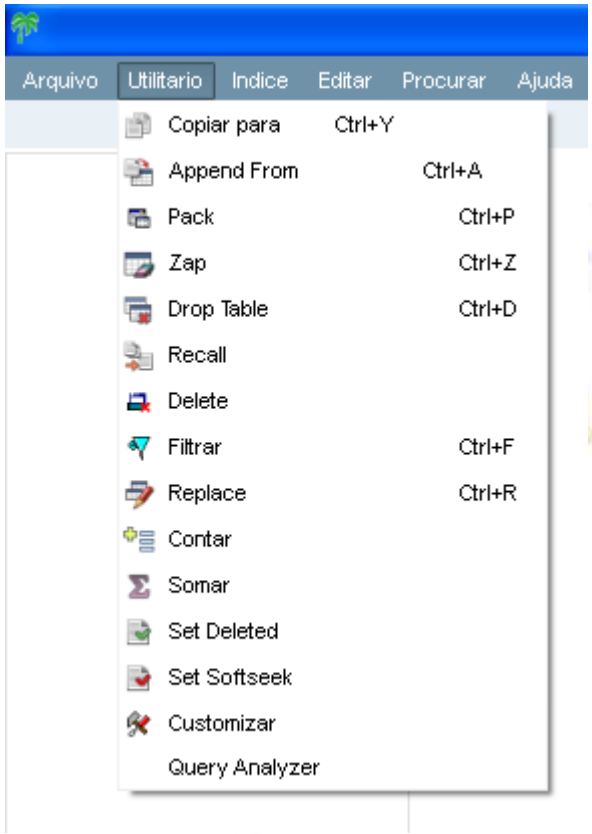


Outras funções



Abrir (Ctrl+B) – Abre tabelas de base de dados (DBF / TOP, etc.
Fechar – Fecha Tabela Corrente
Estrutura - Mostra estrutura da Tabela Corrente
Nova Estrutura – Cria nova estrutura
Posição dos Campos – Mostra posição dos campos e permite mudar a ordem
Status – Informações da tabela corrente
Importar (Ctrl +T) – Importa dados de base de dados, utilizado também para converter tipos de dados diferentes.
Sair (Alt+F4) – Sai do MPSDU.

APSDU / MPSDU



Copiar para (Ctrl+Y) – Copia tabelas entre base de dados.

Append From (Ctrl+A) – Inserir Registro de Base Dados escolhida que contenha alguns campos iguais a tabela corrente.

Pack (Ctrl+P) - Elimina os registros marcados como deletados da tabela corrente, de acordo com condição especificada.

Zap (Ctrl+Z) – Exclui definitivamente todos os dados da tabela corrente.

Drop Table (Ctrl+D)– Exclui definitivamente a tabela do banco, apenas para TOP.

Recall – Permite desmarcar os Registros da tabela corrente que foram marcados como Deletados

Delete – Marca registros como deletados, de acordo com condição especificada.

Filtrar (Ctrl+F) – Filtra a tabela corrente de acordo com condição especificada.

Replace(Ctrl+R) – Substitui o conteudo do campo especificado de acordo com condições especificadas na tabela corrente.

Contar – Faz contagem de registros da tabela corrente, conforme condições especificadas.

Somar – Soma o campos escohido da tabela corrente, conforme condições especificadas.

Set Deleted – Define exibição de registros deletados ou não.

Set Softseek – Define a utilização de posicionamento exato ou parecido.

Customizar – Customiza botões.

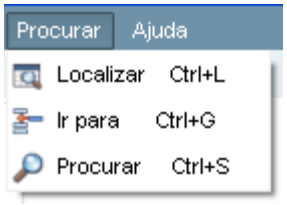
Query Analyzer – Editor de instruções em SQL, apenas para ambiente TOP

APSDU / MPSDU

Inserir – Inserir registros na tabela corrente.

Alterar – Alterar registro atual na tabela corrente.

Apagar – Apagar registro atual na tabela corrente.



Localizar (Ctrl+L) – Localiza registros na tabela corrente, de acordo com condições definidas.

Ir para (Ctrl+G) – Posiciona exatamente no registro especificado.

Procurar (Ctrl+S) – Posiciona no registro especificado de acordo com a chave da ordem corrente.

Telas padrão Microsiga

Telas padrão Microsiga

mBrowse(nTop, nLeft, nBot, nRig, cAlias)

Modelo2()

Modelo3()

Telas padrão Microsiga

Premissas para o Browse

criar variavel private cCadastro do tipo Caracter

criar variavel private aRotina do tipo Array

a chamada da funcao mbrowse envia para a funcao 3 argumentos: -

alias()

recno()

Opcao do browse

A funcao deve receber esses argumentos:

EX: funcao(cAlias, nReg, nOpc)

Telas padrão Microsiga

Fazer um browse pelo assistente de código e deixar as opçoes:

Pesquisa (axpesqui)

Visualiza(axvisual)

GAIA (colocar a user function do exercicio da página 20)

Telas padrão Microsiga

Diferença:

Modelo2 – Monta tela com capa e item, onde o Alias para capa e item é o mesmo

Modelo3 – Monta tela com capa e item, onde o Alias para capa é um e para item é outro.

Telas padrão Microsiga

Passos para construção

Criar variável private aHeader do tipo Array

Criar variavel private aCols do tipo Array

Preparar variáveis de memoria dos campos de acordo com o dicionário

Montar aHeader de acordo com o dicionário

Montar aCols de acordo com o dicionário

Chamar a funcao modelo3

Telas padrão Microsiga

Criar variável private aHeader e Acols

```
aHeader := {}
aCols   := {}
```

Preparar variáveis de memória dos campos de acordo com o dicionário

Regtomemory(cAlias, lInclui)

Telas padrão Microsiga

Monta aHeader

```
SX3->(dbSetorder(1))
SX3->(dbSeek("SC6" ))
While SX3-> (!EOF() .AND. SX3->X3_ARQUIVO == "SC6")
      If X3USO(SX3->X3_USADO) .AND. cNIVEL >= SX3->X3_NIVEL
            AAdd (aHeader, {Trim(SX3->X3_TITULO),;
                                    SX3->X3_CAMPO       ,;
                                     SX3->X3_PICTURE     ,;
                                    SX3->X3_TAMANHO     ,;
                                    SX3->X3_DECIMAL     ,;
                                    SX3->X3_VALID       ,;
                                    SX3->X3_USADO       ,;
                                    SX3->X3_TIPO        ,;
                                    SX3->X3_ARQUIVO     ,;
                                    SX3->X3_CONTEXTO})
      ENDIF
       SX3->(DBSKIP())
ENDDO
```

Telas padrão Microsiga

Monta aCols

```
Aadd (aCols, Array(Len(aHeader)+1))

For nAdi:= 1 to Len(aHeader)
   aCols[1] [nADI] := Criavar(aHeader[nADI][2])
Next

ACols[1][len(aHeader)+1] := .F.
```

Telas padrão Microsiga

Chama a função modelo3

lRet := Modelo3(cCadastro, "SC5", "SC6", ,“Allwaystrue",“Allwaystrue" , nOpc, nOpc)

Telas padrão Microsiga

Sintaxe

Modelo3(cCadastro, cAlias1, cAlias2, [aCpoEnchoice], cLinOK, ;
cTudoOK, nOpcE, nOpcG, [cFieldOK,] [lVirtual], ;
[nLinhas], [aAltEnch], [nFreeze],[aBut], [aCordw], [nSizeHead])

cCadastro – Titulo da Janela
cAlias1 – alias da enchoice
cAlias2 – Alias da getdados
aCpoEnchoice – Campos da enchoice, Se nulo obedece dicionario
cLinOk - Validacao da linha na Getdados
cTudook – Validacao no Ok
nOpcE - Opcao da Enchoice
nOpcG – Opcao da Getdados
cFieldOk – Validacao dos campos da enchoice, Se nulo obedece dicionario
lVirtual – Aceita campos Virtuais , Se nulo .T.
nLinhas – Qtde de Linhas da Acols, , Se nulo, qtde de linhas ilimitadas
aAltEnch - Campos que podem ser alterados na enchoice, Se nulo obedece dicionario
nFreeze – Campos que devem ser congelados
aBut – Botões a serem inseridos na enchoicebar
aCordW – Array com as coordenadas da janela, Ex: {1,5,580,380}
nSizeHead – Qtde de linhas para a enchoice

Telas padrão Microsiga

Exercício modelo3

Amarração Cliente x Produto
Tabelas Envolvidas
- Cadastro de Clientes (SA1) , sempre visual (não permite alteracao)
- Amarracao Produtos x Clientes (SA7) Obedece a escolha do usuario

Browse (Cadastro de Clientes)
- Itens do aRotina
- Incluir, Alterar ,visualizar , excluir os dados
  - Pesquisa Padrão (AXPESQUI)
  - Não permitir incluir o mesmo produto para o mesmo cliente.
  - Não permitir alteração caso não exista nenhum item amarrado para o cliente selecionado.
  - Na exclusão, excluir apenas os itens (SA7), Não deixar excluir caso não exista nenhum item amarrado para o cliente selecionado.

-Trazer o acols já preenchido conforme registro posicionado pelo usuário.

Funções de Servidor

Funções de Servidor

GetRemoteIniName()-
Retorna o nome do arquivo de configuração remote.

GetEnvServer() -
Retorna o ambiente atual

GetsrvProfString("ROOTPATH", "C:\")
Retorna o conteúdo do parâmetro da seção

Exemplo:

```
cTemProc := "Menu: " + CARQMNU + CRLF //Nome do menu
cTemProc += "Modulo: " + cModulo+ CRLF // Modulo atual
cTemProc += "Ambiente: " + Getenvserver() + CRLF //Ambiente atual
cTemProc += "Usuario: " + cUserName + CRLF //Nome Usuário
cTemProc += "Programa: " + funname() + CRLF //Funcao chamada Menu
```

Funções de Servidor

Funname(n)- Retorna a função da pilha, onde n é a posição na pilha.

Conout(cMens) – Imprime cMens no console do Server

GetFuncArr(“A”) – Retorna as funções do repositório.

GetFuncPrm(cFun) – Retorna parâmetros da função especificada.

GetTheme() – Retorna o tema escolhido.

GetDBExtension() – Retorna a extensão da Base (DBF, DTC, etc)

OrdBagExt() – Retorna a extensão do Indice(CDX, IDX, NTX, etc)

Cópia Cliente / Servidor

CPYS2T - Copia arquivos do servidor para a máquina do cliente.

CPYS2T(cArqOri, cArqDes, lComp)

cArqOri – Arquivo de Origem, visivel a partir do rootpath do ambiente corrente

cArqDest – Arquivo de Destino, visivel na máquina do cliente

lComp – Define se compacta antes de enviar o arquivo.

CPYT2S - Copia arquivos da máquina do cliente para o servidor.

CPYT2S(cArqOri, cArqDes, lComp)

cArqOri – Arquivo de Destino, visivel na máquina do cliente

cArqDest – Arquivo de Origem, visivel a partir do rootpath do ambiente corrente

lComp – Define se compacta antes de enviar o arquivo.

Cópia Cliente / Servidor

Exemplo:

PREPARE ENVIRONMENT EMPRESA '99' FILIAL '01' MODULO 'FAT‘;

TABLES "SA1", "SA2","SA3"

Ou

RpcClearEnv() //LIMPA O AMBIENTE

RpcSetType(3) //Nao utiliza licenca

RpcSetEnv("99","01",cUser,cSenha,"FAT",,{"SA1", "SA2","SA3"})

Rotinas Automáticas

MSEXECAUTO

Objetivo

Fazer manutenção automática (inclusão, alteração e exclusão) das rotinas de manipulação de dados do sistema, automatizando o processo de entrada de dados sem a necessidade de desenvolver rotinas especificas.

MSEXECAUTO

Vantagens

1. Não precisa de interface;
2. Segurança;
3. Agilidade na customização.

Simula a digitação pelo usuário, portanto todos os gatilhos, validações e pontos de entrada serão executados.

MSEXECAUTO

Sintaxe:

MSExecAuto({|x,y|fpgmpad(x,y)},aCampos,nOpc)

- fPgmPad = Nome da Função Padrão
- aCampos = Array com os campos a gravar
- nOpc = Opção que será utilizada.

MSEXECAUTO

Exemplo:

```
User Function fCadForn()
Local aVetor := {}
lMsErroAuto := .F.
aVetor := {   {"A2_COD"    ,"999999"                    ,nil},;
              {"A2_LOJA"   ,"01"                        ,nil},;
              {"A2_NOME"   ,"Fornecedor Teste"  ,nil},;
              {"A2_NREDUZ" ,"Teste F"                   ,nil},;
              {"A2_END"    ,"Rua teste"                 ,nil},;
              {"A2_MUN"    ,"teste"                     ,nil},;
              {"A2_EST"    ,"SP"                        ,nil}}
MSExecAuto({|x,y| Mata020(x,y)},aVetor,3) //Inclusao
If lMsErroAuto
        Alert("Erro")
        mostraerro()
Else
        Alert("Ok")
Endif
Return
```

MSEXECAUTO

EXERCÍCIOS

Criar uma rotina que inclua automaticamente pelo MSEXECAUTO um pedido de vendas com 2 itens.
Coloque essa rotina no Browse do Exercício anterior.

Apoio: ROT_AUTO.PRW
Funcao de inclusao:mata410()
Tabelas: SC5 e SC6

MSEXECAUTO

EXERCÍCIOS

Criar uma rotina automática para o cadastro de Clientes, para isso crie um grupo de perguntas com as seguintes perguntas:

Nome, Endereço, CNPJ, telefone

Use o MSExecauto com esses campos.
Coloque no aRotina do Browse do exercício anterior.
Lembre-se de gravar o código usando a função
GETSXENUM()
Mostre na tela o código gerado