02 Criacäo de Variaveis

Revisão: 30/09/2002

Abrangência

Versão 5.07	Versão 5.08	Versão 6.09	Versão 7.10	Versões Anteriores

Na criação de uma variável deve-se ter em mente alguns pontos fundamentais:

- A declaração
- O tipo de variável
- A função CRIAVAR()
- A inicialização
- Padronização de variáveis

A Declaração

Deve ser feita sempre no início da rotina que for utilizá-la, como no exemplo:

```
Function a910VerCod()
Local cCod910 := “001”
Return
```

O Tipo de Variável

O tipo de variável serve para identificar a utilização que a mesma terá no decorrer da rotina. Toda variável deve estar tipada durante sua criação. Quando programamos nativamente em “C”, isto se torna obrigatório. Devemos fazer o mesmo no AP5, pois isto demonstra que a variável foi conscientemente declarada.

Tipos Existentes

PUBLIC: Esta variável será inicializada em um valor lógico falso (.F.) até que seja atribuído um valor específico a ela. Esta variável permanece definida por toda a duração da aplicação e pode ser vista (assim como usada, alterada e avaliada) por qualquer função. Esta variável gera um token (indicação) na tabela de símbolos, isto significa que o módulo principal conterá símbolos para esta classe de variável, o que, por sua vez, ocupa mais espaço de memória. Deve-se evitar a utilização deste tipo, a não ser em casos extremos.

PRIVATE: Esta variável será inicializada em valor nulo (NIL) e uma vez declarada, permanecerá assim durante toda a duração do fluxo da função, até que este volte ao procedimento inicial que a chamou. Em essência, uma variável de memória PRIVATE inicializada logo no início do Protheus, agirá como um variável PUBLIC. Esta variável pode ser vista por uma sub-rotina da função e modificada de maneira correspondente. Esta variável também gera um token na tabela de símbolos comentada acima.

LOCAL: Esta variável de memória será inicializada com valor nulo (NIL) e só é visível dentro da função que a inicializa, mesmo que esta última, contenha funções incorporadas a seu conteúdo. Este tipo de variável é o mais adequado a ser utilizado em funções, pois não gera símbolos na tabela de símbolos, por conseqüência ocupa pouco espaço de memória e, o compilador avalia as variáveis LOCAL e STATIC mais rapidamente que os outros tipos (PUBLIC e PRIVATE). Cuidado para não sucumbir à teoria de que se pode obter economia de memória, mudando qualquer referência PRIVATE para uma referência LOCAL. Se você fizer isso, as funções podem não funcionar corretamente, embora funcionassem na versão anterior às alterações.

STATIC: A variável STATIC é idêntica à classe de armazenamento LOCAL, com uma exceção. Uma variável STATIC é retida dentro de sua sub-rotina, mesmo depois que o fluxo da função a tenha deixado. Isto é particularmente útil para funções independentes tipo “caixa-preta”, que contém seu próprio conjunto de variáveis exclusivas e devem manter esses valores de interação em interação.

Inicialização

Quando não atribuímos nenhum valor a uma variável no momento de sua declaração, corremos o risco de utilizá-la com valor “NIL” e causar erros fatais. Por isso, a inicialização de uma variável é de extrema importância.

Padronização de Variáveis

É importante que ao lermos o nome de uma variável, possamos saber se o seu tipo é numérico, caracter, data ou lógico. O nome da variável de get não deve coincidir com uma variável de outro programa, pois toda variável de get possui um help específico.

Exemplo:

a variável DBaixa (get da baixa no programa de Títulos a Receber), já possui um texto help que indica seu conteudo e não deverá ser criada outra variável para outra finalidade com este mesmo nome.

Para tanto, definimos a seguinte padronização :

```
N -> Numéricas
L -> Lógicas
D -> Data
C -> Caracter
A -> Array (matriz)
O -> Objeto
U -> Sem definição
```

Criando uma Variável Utilizando a Função CRIAVAR()

Esta função cria uma variável, retornando o valor do campo, de acordo com o dicionário de dados. Avalia o inicializador padrão e retorna o conteúdo de acordo com o tipo de dado definido no dicionário.

Sintaxe

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

Onde :

```
Uret -> tipo de retorno de acordo com o dicionário de dados, considerando inicializador padrão.
```

cCampo -> Nome do campo

LiniPad -> Indica se considera (.T.) ou não (.F.) o inicializador padrao (X3_RELACAO)

Clado -> Lado para inicializaçäo padrão

Variáveis de Relatórios

Na criação de um relatório algumas variáveis e seus tipos são convencionados para a utilização da biblioteca de funções de relatório.

Variável	Tipo	Conteúdo
wnRel	Local	Nome default do relatório em disco
cbCont	Local	Contador
Cabec1	Local	1ª linha do cabeçalho do relatório
Cabec2	Local	2ª linha do cabeçalho do relatório
Cabec3	Local	3ª linha do cabeçalho do relatório
Tamanho	Local	Tamanho do Relatório (P = Pequeno 80 colunas, M = Médio 132 colunas, G = Grande, 220 colunas)
cDesc1	Local	1ª linha da descrição do relatório
cDesc2	Local	2ª linha da descrição do relatório
cDesc3	Local	3ª linha da descrição do relatório
Limite	Local	Quantidade de colunas no relatório (80,132,220)
Titulo	Local	Título do Relatório
aReturn	Private	Matriz com as informações para a tela de configuração de impressão
Nomeprog	Private	Nome do programa do relatório
cString	Private	Alias do arquivo principal do relatório para o uso de filtro
Li	Private	Controle das linhas de impressão. Seu valor inicial é a quantidade máxima de linhas por página utilizada no relatório

m_pag	Private	Controle do número de páginas do relatório
aOrd	Private	Matriz contendo as ordens de layout para a impressão. Caso não existam várias ordens esta matriz deve estar vazia. Ex.: aOrd := {"Código", "Descrição", "Telefone"} -> O layout do relatório vai depender da ordem selecionada na tela de configuração de impressão
nLastKey	Private	Utilizado para controlar o cancelamento da impressão do relatório
cPerg	Private	Nome da pergunta a ser exibida para o usuário
aLinha	Private	Matriz que contem informações para impressão de relatórios cadastrais

Grupos Relacionados

Principal / Guias de Referência / Como programar Advpl no ERP

Topo da Página