INDICE

INTRODUÇÃO

90 Dicas de Visual Basic

Suplemento Especial - As figurinhas que faltavam

"Bafo!" Gritou o garotinho de oito anos, ao trocar uma figurinha com o colega. "No meu álbum falta a número trinta, você tem?". Anos depois, continuamos a trocar figurinhas. Por exemplo: "Você sabe como criar um servidor OLE?", "Como acesso um banco ODBC no VB?", e outras figurinhas. Nesta edição, trazemos algumas delas.

Convenção: *VB3*/*VB4* = versão aplicável à dica. *Por* = autor. *Aperf*. = adaptado e aperfeiçoado por.

1 - VB4 - Atalhos para VB no Windows 95

Com a versão quatro do Visual Basic e o novo ambiente de sistema operacional de 32 bits, eu usava três versões do VB. Alguns de meus clientes não aceitavam aplicações construídas em VB4. E alguns não migraram para 32 bits. Após instalar ambas as versões 16 e 32 bits em minha máquina com Windows 95, descobri que qualquer projeto com extensão .VBP pode ser aberto no VB 32 bits. Este é o melhor caminho para executar a correta versão do VB:

1) Salve todos os arquivos de um projeto na mesma pasta.
2) Crie um atalho para o VB, na edição que você usa.
3) Arraste o projeto (VBP) para o topo do atalho e (*tcham*); o projeto será aberto por esta versão.

*Por Joe Sytniak**

2 - VB3/VB4 - Criando um procedimento de pausa

Falta um comando do VB para provocar uma pausa (*wait, dalay, pause* etc.) no processamento? Basta implementar uma pequena rotina, em um módulo (.BAS):

```
Function FU_Delay (Quanto As Double, PermiteDoEvents As Integer) As Double
   'executa uma pausa na aplicação
   'quanto = tempo da pausa (em segundos)
   '    pode ter frações de segundos
   'PermiteDoEvents é true ou false
   '     DoEvents permite realizar outras tasks do Windows
   'Timer é uma função do VB que retorna
   '    o nr. de segundos desde meia noite
   ' RETORNO: o tempo de looping
   '    que devido a imprecisão e multitask, pode ser
   '    diferente do valor pedido
   Dim Inicio As Double
   Dim Check  As Double
   Dim Contador As Double
   Contador = Timer
   Inicio = Timer
   Do Until Check >= (Inicio + Quanto)
      Check = Timer
      If PermiteDoEvents Then DoEvents
   Loop
   'o VB dá uma boa precisão em 1/10 de segundo
   'a 1/100 a precisão já é parcialmente comprometida
   'a 1/100 a precisão se perde
   FU_Delay = (Timer - Contador)
End Function
```

Note que **não** foi necessário inserir um controle *Timer* no form para executar esta pausa.
Para acessar a rotina basta citar o nome e passar o número de segundos. O segundo parâmetro informa se deve ser usado *DoEvents*.
vPausa = FU_Delay (4.5, False) ‘pausa de 4 segundos e meio
O valor de *vPausa* poderá não ser 4.5 (uma subtração poderá servir de teste). O Windows executa várias tarefas ao mesmo tempo. Assim, uma tarefa poderá não ser executada duas vezes com o mesmo tempo. A chamada ao *DoEvents* provoca uma melhoria na distribuição de tarefas, para que a pausa não atrapalhe os demais programas.
Se desejar criar um procedimento de pausa mais simples, sem tanta preocupação com a precisão, a função poderá ser chamada por outra rotina, do tipo *sub*. Abaixo, esta rotina toma o *DoEvents* como ativado e simplifica a sintaxe.

```
SU_Delay 4.5 'chamada a uma pausa, como se fosse um comando
Sub SU_Delay (quanto as double)
        dim ret as double
        ret = FU_Delay(quanto - .01, true)
End Sub
```

Por Charles A. Müller.

3 - VB3/VB4 - Não vá embora sem avisar

Usuários podem, por descuido, sair da sua aplicação através **da Lista de Tarefas** ou **Barra de Tarefas**, ou ainda, saindo do Windows. Adicione um procedimento ao evento *QueryUnload* do form principal para prevenir o problema.
Este evento possui um parâmetro, o *UnloadMode*, que permite detectar como o fechamento do form foi invocado. Se o valor for 1, representa que o fechamento ocorreu via código (comando *Unload*). Se o valor do parâmetro *cancel* for alterado para *true*, o fechamento do form é cancelado. Veja *QueryUnload* e *Using MDI Features* no Help do VB para maiores detalhes.

```
Private Sub Form_QueryUnload (Cancel As Integer, UnloadMode As Integer)
        If UnLoadMode <> 1 then
                If 7 = MsgBox("Deseja realmente sair do sistema?", 32 + 4) Then
                        'respondeu não
                        Cancel = True
                End If
        End If
End Sub
```

O código acima (VB4) também se aplica ao VB3.
Por Jiyang Keven Luo, aperf. por Charles A. Müller*

4 - VB3/VB4 - Programando de forma diferente em tempo de desenho e execução

Este código habilita ou desabilita funções durante o desenho e teste. O código poderá permanecer durante o desenvolvimento, sem afetar o usuário final. Verifique se o caminho procurado é o caminho do seu projeto e não o diretório final de sua aplicação.

```
If InsStr(App.Path, "VB") Then
        'execute os processos próprios
        'de debug e não de sistema executável
End IF
```

Uma variação é:

```
If InsStr(App.Path, "VB") Then Stop
```

Você pode inserir este código para depuração (debug); se você esquecer, isto não causará - repetimos - problemas ao usuário.

*Por John Bailey **

5 - VB4 - Novas funções de Registry

Há uma lista de novas funções do VB4 para trabalhar com entradas do Registry (ou INI, em plataformas Windows de 16 bits): *GetSetting, GetAllSettings, SaveSetting e DeleteSetting*. Estas novas funções eliminam a necessidade de chamadas API. Veja no VB Help maiores detalhes procurando "*Additional Information on VBA Registry Functions*".

*Por Denis Basaric e Norbert Steinhoefel-Carqueville**

6 - VB3 - Carregando forms do VB4 no VB3

Você não poderá ler um form do VB4 diretamente no VB3. A definição do form deve ser alterada em um editor de texto (ASCII).

```
VERSION 4.00
Begin VB.Form Form1
        Caption = "Form1"
        ClientHeight = 5940
        'demais propriedades
        '...
End
Attribute VB_Name = "Form1"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
Private Sub Form_Load()
        '...
End Sub
```

Mude a versão 4.00 para VERSION 2.00, remova todos os sufixos *VB.* no comando *Begin Form*, remova todas as declarações *Attribute*. Remova também a cláusula *Private* de algumas rotinas (eventos). Salve o arquivo e abra-o no VB3.
Por Saji Varghese, aperf. por Charles A. Müller*

7 - VB3/VB4 - Como calcular as coordenadas (x,y) de qualquer posição de um círculo

A rotina abaixo (parte da biblioteca *CodeBank*) calcula as coordenadas de qualquer ponto, medida em graus, numa circunferência, num círculo ou numa elipse. Como você pode notar, é uma rotina simples, mas extremamente útil no desenho de gráficos ou movimentação de objetos.

```
Public Sub DegreesToXY(CenterX as Long, CenterY as Long, _
        Degree as Double,     RadiusX as Long, RadiusY as Long, _
        X as Long, Y as Long)
        Dim Convert as Double
        Convert = 3.141593 /180 ' PI/180
        X = CenterX - (Sin(-Degree * Convert) * RadiusX)
        Y = CenterY - (Sin((90 + Degree) * Convert) * RadiusX)
End Sub
```

Por Ward Hitt, autor do Visual Components Inc.'s CodeBank

8 - VB3/VB4 - Procurando por nulos retornados por chamadas DLL

Após uma chamada a DLL (API), o valor retornado pode conter um nulo. Um dos meios de eliminar este nulo é procurar o caracter Chr$(0), como neste exemplo.

```
'yourstring é a string retornada pela API, e pode conter um nulo
Dim CheckForNull as Integer
CheckForNull = Instr(YourString, Chr$(0))
If CheckForNull > 0 then Left$(YourString, CheckForNull - 1)
```

Por Marc Mercuri

9 - VB4 - Erros de Licença

Enfrentei um interessante problema tentando instalar a edição *Enterprise* do VB4 no Windows 3.1. A nova versão do VB usa o *Registry*, que no Windows 3.1 é limitada a 64Kb. Como consultor de uma grande empresa, tenho muitos softwares instalados no meu laptop para trabalhar em vários ambientes de cliente. Já havia instalado MS Office, MS Project e Lotus Suite Standart. O arquivo REG.DAT já estava cheio.
Quando instalei o VB4, não foram indicados erros de instalação. Mas, ao tentar usá-lo, juntamente com alguns controles, surgiram erros de "licença".

Liguei para a Microsoft e recebi a seguinte orientação:
1. Remover manualmente o VB4.
2. Remover manualmente todos os OCXs e OCAs do diretório Windows/System.
3. Remover manualmente a OC25.DLL do Windows/System.
4. Renomear REG.DAT para REG.OLD.
5. Remover todos os itens do grupo *Start Up* (Iniciar).
6. Remover as entradas *Load* e *Run* em WIN.INI.
7. Remover todos os TSRs de AUTOEXEC.BAT.
8. Se você utiliza um drive compactado, libere 6MB de espaço em um volume não compactado.
9. "Resete" o micro.
10. Reinicie o Windows e reinstale o VB4.
11. Redefina as opções de sistema.

Nota do VBPJ:* Se estes erros de licença ocorrerem no Windows 95, remova e reinstale o VB4 para corrigir o problema.
*Por Jim Gilligan**

10 - VB3/VB4 - Valores de retorno não requeridos

Você não precisa retornar valores em todas as funções. Mas, é uma implementação um pouco perigosa.

```
Private Sub Form_Load( )
        dice
End Sub
Function dice ( ) As Integer
        dice = Int(Rnd * 6) + 1
        MsgBox "Esta é uma rotina que não retorna valor"
End Function
```

Nota da Redação: Esta implementação apenas é útil em empresas que padronizam todo o código para funções. Recomendamos o uso de ***sub*** e não de *function*, para um procedimento que não retorna valor *(Charles A. Müller).*
*Dica de Andy Rosa**

11 - VB4 - Atualizando Bound Controls por uma List Box

Quando você desejar que os *Bound Controls* (controles associados a dados) sejam atualizados em eventos de listas ou combos, adicione este código no evento *click* (ou double-click) da lista ou combo:

```
Data1.RecordSet.Bookmark = DBCombo1.SelectedItem
```

Como resultado, seu registro corrente passará a ser o registro com a chave indicada na lista ou combo. Todos os *Bound Controls* são atualizados automaticamente. É necessário definir apenas as propriedades *RowSource* e *ListField*. Assim, economiza-se tempo que seria gasto em conversões de dados e atualização de campos.
*Por Peter Klein**

12 - VB4 - Destacando uma linha em um DBGrid

Para destacar uma linha no controle DBGrid, adicione o registro corrente à *SellBookmarks Collection*:

```
Private Sub DBGrid_RowColChange _
        (LatRow As Variant, ByVal LasRow As Integer)
        If Data1.RecordSet.RecordCount Then
                DBGrid.SelBookmarks.Add _
                        Data1.RecordSet.Bookmark
        End If
End Sub
```

*Por Peter Chyan**

13 - VB3/VB4 - Objetos vazios?

Não se pode usar a função *IsEmpty* para determinar se uma variável-objeto (como Form ou qualquer controle) possui valor. É possível, entretanto, usar a implementação abaixo para determinar se uma variável de form (ou outro objeto) está vazia.

```
If Not frmChild Is Nothing Then
        Unload frmChild
End If
```

*Por Arn Cota**

14 - VB3/VB4 - Livre-se dos zeros inúteis

Vamos retirar os zeros inúteis da variável mystring (que contém "00030"). Abaixo, um interessante caminho para isto.

```
Mystring = CStr(CInt(mystring))
```

Outro caminho é:

```
Mystring = Str(Val(mystring))
```

Por Brad Herbert aperf. por Charles A. Müller*

15 - VB3/VB4 - Campos na peneira

Muitas vezes, utiliza-se um campo formatado para exibição, e se grava um valor "peneirado", ou seja, de um formato específico. As funções abaixo "limpam" strings de números ou alfabéticos. Esta é uma alternativa ao controle *Masked Edit*.

```
Function FU_LimpaNumero (campo As String) As String
   'recebe string numérica
   'retorna string numérica sem pontos, vírgulas etc.
```

```
   'exemplo FU_LimpaNumero("1.245,90") = "1234590"
   Dim VA_Posicao As Integer
   Dim VA_Caracter As String * 1
   Dim VA_Resultado As String
   VA_Resultado = ""
   VA_Posicao = 1
   Do While VA_Posicao <= Len(campo)
      VA_Caracter = Mid$(campo, VA_Posicao, 1)
      If IsNumeric(VA_Caracter) Then
         VA_Resultado = VA_Resultado & VA_Caracter
      End If
      VA_Posicao = VA_Posicao + 1
   Loop
   FU_LimpaNumero = VA_Resultado
End Function
Function FU_LimpaAlfa (campo As String) As String
   'recebe string alfanumérica
   'retorna string de letras maiúsculas  sem pontos, vírgulas, números etc.
   'exemplo FU_LimpaNumero("Adq-7465") = "ADQ"
   Dim VA_Posicao As Integer
   Dim VA_Caracter As String * 1
   Dim VA_Resultado As String
   VA_Resultado = ""
   VA_Posicao = 1
   campo = UCase(campo)
   Do While VA_Posicao <= Len(campo)
      VA_Caracter = Mid$(campo, VA_Posicao, 1)
      If Asc(VA_Caracter) > 64 And Asc(VA_Caracter) < 91 Then
         VA_Resultado = VA_Resultado & VA_Caracter
      End If
      VA_Posicao = VA_Posicao + 1
   Loop
   FU_LimpaAlfa = VA_Resultado
End Function
```

Por Charles A. Müller

16 - VB3/VB4 - Convertendo Identificadores em Rótulos e Cabeçalhos

Programadores possuem o hábito de criar identificadores (nomes de variáveis, por exemplo) por fusão de palavras como *SobreNome* ou *CargoAnterior*. É possível usar alguns destes nomes para se criar *labels* (rótulos) e descrições diversas. A função abaixo insere espaços, "quebrando" os identificadores a cada inicial maiúscula. Assim, *CargoAnterior* será convertido para *Cargo Anterior*.

```
Function SpaceName (src As String) As String
        Dim i as Integer, tgt As String
        tgt = Left$(src,1)
        For i = 2 to Len(src)
```

```
                Select Case Mid$(src, i-1, 1
                        Case "a" to "z"
                                Select Case Mid$ (src, i, 1)
                                        Case "A" to "Z"
                                                tgt = tgt & " "
                                End Select
                End Select
                tgt = tgt & Mid$(src, i, 1)
        Next i
        SpaceName = tgt
End Function
```

*Por Pat Dooley**

17 - VB3/VB4 - Alterações com Mid

Você provavelmente já conhece a função e o comando Mid, que retorna uma substring com um número específico de caracteres, ou seja, uma parte da string usada como parâmetro. Mas, você sabe como usar o Mid para **substituir caracteres no meio** de uma string? O Mid é uma pequena excentricidade do VB, pois, altera um de seus próprios argumentos. Mas, isto **economiza uma série de instruções de concatenação**, observe:

```
Dim mystring as String
mystring = "SOME STRING"
If Mid(mystring, 2, 1) = "O" Then
        Mid(mystring, 2, 1) = "A" ' substituindo caracter
End If
```

*Por William Storage**

18 - VB3/VB4 - Quando usar SendKeys

A função *SendKeys* (que simula o aperto de teclas) adiciona ótimos recursos de "intervenção" do programador na operação do sistema. As teclas podem ser enviadas para um form ou controle (neste caso o controle deverá ter o foco). A rotina abaixo simplifica o processo.

```
Sub SendKeyTo (KeyValue as String, cCnt as Control)
        If cCnt.Enabled Then cCnt.SetFocus
        SendKeys KeyValue
End Sub
```

Por Saji Varghese, aperf. por Charles A. Müller

19 - VB3/VB4 - Resolução do Monitor

Há uma forma simples de se obter a resolução de um monitor de vídeo usando uma API.

```
'declarations
Declare Function GetSystemMetrics Lib "User" (ByVal nIndex As Integer) as Integer
'...
Sub Form_Resize( )
        dim xRes As Integer
        dim yRes As Integer
        xRes = GetSystemMetrics(0)
        yRes = GetSystemMetrics(1)

        If  xRes < 1024 and yRes < 768 Then
                'adicione seu código de controle de dimensões
        End If
End Sub
```

*Por Sanjay Mawalkar**

20 - VB3/VB4 - Fechando todos os forms

```
Sub UnloadAll ( )
        Dim f As Integer
        f = Forms.Count
        Do While f > 0
                Unload Forms(f-1)
                If f = Forms.Count Then Exit Do
                f = f - 1
        Loop
End Sub
```

*Por Denis Basaric**

21 - VB4 - Subclasse para ChDir

Se o seu diretório de aplicação é D:\OldDir, a chamada ChDir(C:\NewDir) irá alterar o drive corrente para o diretório NewDir. Mas o diretório da aplicação continuará sendo D:\OldDir. Causa: o ChDir altera o diretório na unidade **diferente** citada e não muda o da aplicação. Esta rotina, de classe subdefinida (*subclassed*), melhora a alteração de drives:

```
Sub ChDir(Path As String)
        Dim TargetDrive As String
        If Mid(Path, 2, 2) = ":\" Then
                TargetDrive = Left(Path, 3)
```

```
            If TargetDrive <> Left(CurDir, 3) Then
                ChDrive TargetDrive
            End If
        End If
        'chama a função ChDir do VB
        VBA.ChDir Path
End Sub
```

*Por Bruce Hamilton, Centric Development**

22 - VB3/VB4 - Graduando Cores

Este é um meio fácil para pintar o fundo de um form, com efeito de uma "cortina degradê", isto é, da cor mais clara no topo, para a cor mais escura na base. Para especificar a cor usada passe valores *true* ou *false* para os parâmetros Vermelho, Verde ou Azul. Combinações de cores (true) formam outras cores como roxo, amarelo, cinza etc.
O código abaixo cria o efeito com a cor azul.

```
FadeForm Me, False, False, True
   'qualform, vermelho?,verde?, azul?
```

Já este cria o efeito com a cor ciano (mistura do azul com verde)

```
FadeForm Me, False, True, True
   'qualform, vermelho?,verde?, azul?
```

O código da rotina segue abaixo

```
Sub FadeForm (frm As Form, pRed As Integer, pGreen As Integer, pBlue As Integer)
   Dim SaveScale As Integer, SaveStyle As Integer, SaveDraw As Integer
   Dim y As Long, x As Long, i As Long, J As Long, pixels As Long
   'salvar as configurações atuais do form
   SaveScale = frm.ScaleMode
   SaveStyle = frm.DrawStyle
   SaveDraw = frm.AutoRedraw
   'pintar a tela
   frm.ScaleMode = 3
   pixels = Screen.Height / Screen.TwipsPerPixelY
   x = pixels / 64 + .5
   frm.DrawStyle = 5
   frm.AutoRedraw = True
   For J = 0 To pixels Step x
      y = 240 - 245 * J / pixels
      If y < 0 Then y = 0
      frm.Line (-2, J - 2)-(Screen.Width + 2, J + x + 3), RGB(-pRed * y, -pGreen * y, -
pBlue * y), BF
   Next J
   'restaura configurações do form
   frm.ScaleMode = SaveScale
   frm.DrawStyle = SaveStyle
   frm.AutoRedraw = SaveDraw
End Sub
```

*Por Timothy L. Birch**

23 - VB3/VB4 - Arquivo Existe?

Uma das formas de testar se um arquivo específico existe é utilizando a função *Dir$*, com a identificação completa do caminho do arquivo. Dir$ irá retornar o exato nome do arquivo (se existir) ou um nulo, se não existir. Por exemplo:

```
If Dir$("C:\MYDIR\MYFILE.TXT") <> "" Then
        'o arquivo existe
Else
        'arquivo não encontrado
End If
```

*Por Chuong Van Huynh**

24 - VB3/VB4 - Tenha uma linha 3D entre um menu pulldown e uma barra de ferramentas

Desenhe um 3DPanel com um tamanho (*height*) de 30. Este tamanho não é fácil de ser desenhado manualmente. Apague a *Caption,* mude o *BevelOuter* para 1 (*inset*), *border* para 1 e *Align* para *Top*. Desenhe a barra de ferramentas e o menu.
*Por Mário Manuel Mourão Coelho**

25 - VB4/VB3 - Providenciando menus específicos de contexto para seus objetos de interface

Muito da facilidade do Windows 95 inicia com o fato de que a interface de seus objetos possui seus próprios menus de contexto, acessados por um simples click no botão direito do mouse. Você, desenvolvedor, poderá criar seus próprios menus de contexto também . O exemplo abaixo mostra como isto funciona numa list box chamada lstSample:

1. Defina o menu de contexto, assim como se define qualquer outro menu, como o menu de help, com seus submenus. Diferente do menu de help, entretanto, este menu de contexto deverá ter a propriedade *visible = false*, para que nunca seja visto pelo usuário na barra de menus pull-down. A caption nunca será vista pelo usuário, mas, deverá ser algo compreensível para o programador como *Context Menu de lstSample*. O nome, neste exemplo, será *mnu_lstSample*. Agora, basta definir os submenus, que irão aparecer sobre o objeto com o click do botão direito do mouse. Por exemplo, crie menus como &Remover Item, Remover &Todos, &Adicionar Item etc.
2. No evento MouseMove do objeto desejado (neste caso, lstSample), invoque o método ***PopupMenu***.

```
Private Sub lstSample_MouseDown (button As Integer, Shift As Integer, _
    X As Single, Y As Single)
    Const vbRightButton = 2 'constante VB4
```

```
    If button And vbRightButton Then
        PopupMenu mnu_lstSample
    End If
End Sub
```

3. Deve, obviamente, ser adicionado o código de cada evento menu_click. Veja no help do VB mais detalhes sobre o método PopupMenu.

Por Hassan Davis, MicroHelp Inc*

Nota da redação: segundo o autor*, esta é uma dica de VB4, mas, **já existe** menu de contexto (método *popup*) no VB3, que funciona no Windows 3.1x (e parcialmente no Windows 3.0). A diferença é que, o método possui menos recursos que no VB4. Há um exemplo no VB3 (SAMPLES/CALLDLLS) que mostra como desenvolver Popup menus através da API *TrackPopupMenu* (biblioteca *User*); é um exemplo do VB2 que não deveria estar na versão 3. No VB3, a constante *vbRightButton* chama-se RIGHT_BUTTON. (*Charles A. Müller*)

26 - VB4 - Use seus próprios menus popup

Em VB4, se você quiser mostrar um menu popup para um texto, um menu de sistema (default) será mostrado primeiro e o seu menu só aparecerá quando o menu default for fechado. Para contornar este problema:

```
Private Sub Text1_MouseDown(Button As Integer, _
        Shift As Integer, X  As Single, Y as Single)
        If Button = 2 Then
                Text1.Enabled = False
                PopupMenu myMenu
                Text1.Enabled = False
        End If
End Sub
```

*Por Mário Coelho**

27 - VB3/VB4 - Criando múltiplos níveis de diretórios

Programas de instalação ou outras aplicações podem solicitar ao usuário para que informe onde deseja instalar arquivos ou efetuar alguma operação. Se o diretório não existir, será criado. Mas, o usuário poderá informar vários subníveis de diretórios inexistentes que devem ser criados. A rotina abaixo cria qualquer diretório, em todos os níveis. Basta informar o drive (como C:) e o caminho (como \MYAPP\MYDIR\SUBDIR) a ser criado (se não existir). Podem ser criados diretórios de nomes longos em VB4 32 bits, mas, em 16 bits (inclusive VB3), os nomes serão truncados para a convenção *8.3*. Você poderá adicionar seu próprio código de manipulação de erros, conforme sua necessidade.

Abaixo temos um exemplo simples da criação de vários níveis de diretórios:

```
Sub CreateLongDir(sDrive as String, sDir as String)
        Dim sBuild As String
        While Instr(2, sDir, "\") > 1
```

```
            sBuild = sBuild & Left(sDir, Instr(2, sDir, "\") - 1)
            sDir = Mid$(sDir, InStr (2, sDir, "\")
            If Dir$(sDrive & Sbuild, 16) = "" Then MkDir sDrive & sBuild
        Wend
End Sub
Sub Test( )
        Call CreateLongDir ("C:", "Test\MyApp\MyDir\Long Directory Name\")
End Sub
```

*Por Jeffrey Renton**

28 - VB3/VB4 - Mova e redimensione controles com precisão

Ao desenhar um form, você pode utilizar mouse e teclado para obter melhor precisão. Esta dica serve também para Access 2 e 7 (95).

A - Quando você desejar alterar o tamanho de um controle:

1. Selecione-o
2. Pressione SHIFT e use as teclas de navegação para alterar o tamanho.

B - Quando você desejar mover um controle:

1. Selecione-o
2. Pressione CTRL e use as teclas de navegação para alterar a posição.

*Por Chris Kunicki, repassada por John Chmela (VB Developer's Network)**

*Nota da Redação: Os autores informam que a dica (*A e B*) se aplica ao VB3, mas, não funciona. Acrescentamos, ainda, alguns dados abaixo.*

C - Evitando acidentes

1. O VB4 possui o recurso de trava (*lock*) de tamanho e posição em tempo de desenho. Selecione o(s) controle(s) e clique no botão "cadeado", na barra de ferramentas.
2. O VB3 não possui o recurso de "cadeado", mas, é possível mover ou selecionar os controles com maior cuidado (para alterar várias propriedades ao mesmo tempo, por exemplo). Basta selecionar, passando o mouse no form, uma **área em volta** dos controles. Isto não se aplica a controles contidos em outros objetos (como painéis, *frames* e *picture boxes*).

D - Maior precisão

Use os valores numéricos de tamanho e posição: *left*, *top*, *height* e *width* - correspondentes a **x**, **y'** (eixo y do topo para baixo) , **h** (altura) e **b** (base), respectivamente - na Janela de Propriedades ou Janela de Código. Esta tarefa é um pouco árdua, então, desenhe o controle com medidas aproximadas para depois, ajustar, via digitação de valores.

Aperf. por Charles A. Müller

29 - VB4 - GetModuleUsage em 32 bits

Encontrei uma solução para o problema, da API *GetModuleUsage* não trabalhar em VB4 a 32 bits. A *TaskID* retornada pela função *Shell* pode ser usada por *AppActivate*. Assim:

```
TaskID = Shell("DOSAPP.exe", vbNormalFocus)
```

```
On Error GoTo finished
While True
        DoEvents
        AppActivate TaskID
Wend
Finished:
        On Error GoTo 0
```

*Por John Muiri, repassada por John Chmela (VB Developer's Network)**

30 - VB3/VB4 - Melhorando as declarações API (I)

Muitas rotinas API são declaradas como função, mas, o valor de retorno não é sempre utilizado. A função *SendMessage*, por exemplo, depende da mensagem enviada, não importando o valor de retorno. Outro exemplo é a função *Shell* (se o objetivo for chamar e não monitorar um programa externo, o retorno não será utilizado).
Ocorrem chamadas assim:

```
Dim dummy As Integer
dummy = SendMessage(Text1.hWnd, WM_PASTE, 0, 0&)
```

A variável só foi necessária por causa da declaração. Uma alternativa, é declarar a função como *Sub* e usar um *alias* (apelido).

```
Declare Sub SUB_SendMessage Lib "User" Alias "SendMessage" (byVal hWnd as _
        Integer , byVal msg as Integer, byVal wParam as Any, byVal lParam As Any)
```

Agora, chame pelo nome declarado e não pelo original:

```
SUB_SendMessage Text1.hWnd, WM_PASTE, 0, 0&
```

Observe que, seu código ficou mais produtivo de ser mantido.
Por Francesco Baleno (texto revisado por Charles A. Müller)*

31 - VB3/VB4 - Melhorando as declarações API (II - a volta do SendMessage)

Quando falava de *SendMessage (veja dica anterior)*, lembrei de um outro truque que pode ser interessante para ser incluído em seus hábitos de programação. Quando uso algumas mensagens em particular, o argumento *lParam* é, na verdade, considerado uma combinação de dois valores (*words*) . A mensagem EM_LINESCROLL pode rolar uma text box multilinha; a primeira word (low word) contém o número de linhas para rolar verticalmente e a segunda (hight word), contém o número de linhas para rolar horizontalmente.

```
'rola uma caixa de texto em "HO" linhas
'horizontalmente e "VE" linhas verticalmente
'obs.: isto não funciona corretamente
longValue& = HO * 65536 + VE
...
SUB_SendMessage Text1.hWnd, EM_LINESCROLL, 0, longValue
```

O código acima não trabalha corretamente se HO for positivo e VE for negativo.
A solução é dividir o número long de lParam em dois, na declaração

```
Declare Sub SUB_SendMessage2 Lib "User" Alias "SendMessage" (byVal hWnd as _
        Integer , byVal msg as Integer, byVal wParam as Any, byVal lParam1%,_
        lParam2)
```

A chamada passa a ser:

```
SUB_SendMessage2 Text1.hWnd, EM_LINESCROLL, 0, HO, VE
```

Este truque funciona, pois um valor long integer na "pilha" corresponde a combinação de dois valores word combinados.

*Por Francesco Balena**

32 - VB3/VB4 - Simplificando chamadas API através de funções próprias

Algumas chamadas à função API (DLL) são bastante complexas. Uma dica é criar uma função de código VB que chama a API. Assim, a complexidade da API só irá aparecer uma vez.

Por exemplo, a função *GetPrivateProfileString* que, captura uma configuração de arquivo INI.

```
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal _
        lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault _
        As String, ByVal lpReturnedString As String, ByVal nSize As Integer, _
        ByVal lpFileName As String) As Integer
```

A chamada da função ficaria assim:

```
Global Const Ini_File = App.path & "\Myapp.INI)
'...
Dim VA_LastUser
'chamada a API para capturar o conteúdo de "lastuser" na seção "options"
On Error GoTo Erro_INI
Dim VL_Sec As String, VL_Key As String, VL_Size As Integer
Dim VL_Return As String, VL_FileName As String
Dim VL_SizeHandle As Integer, VL_Valid As Integer
Dim Va_Msg As String
Const CL_Default = "" 'retorno no caso de não encontrar
VL_Sec = "options"
VL_Key = "lastuser"
VL_Size = 30
VL_Return = Space$(VL_Size) 'string  a retornar
VL_SizeHandle = Len(VL_Return) 'tamanho da string de retorno
VL_FileName = Ini_File 'arquivo no formato INI
VL_Valid = GetPrivateProfileString(VL_Sec, VL_Key, CL_Default, _
        VL_Return, VL_SizeHandle, VL_FileName)
 VA_LastUser = Left$(VL_Return, VL_Valid)
   Exit Function 'ou Exit Sub
Erro_LeMeuINI:
 VA_LastUser = CL_Default
```

Nota-se uma complexa e grande quantidade de código. A API não retorna a string procurada e sim um *buffer*. O conteúdo é retornado por um argumento (!) e precisa ser formatado com o tamanho do buffer (função *left*). Para eliminar todo este código a cada necessidade (cada campo INI) foi implementada uma chamada assim:

```
Global Const Ini_File = App.path & "\Myapp.INI)
'...
Dim VA_LastUser
'chamada a API para capturar o conteúdo de "lastuser" na seção "options"
Global Const Ini_File = "MYAPP.INI"
...
VA_LastUser = FU_Le_MeuINI ("options", "lastuser", 30)
```

Abaixo, um exemplo de função "tradutora" de API:

```
Function FU_Le_MeuIni (VL_Sec As String, VL_Key As String, VL_Size As Integer) As
String
   'recebe nome da seção e do parágrafo e tamanho da string de retorno
   'retorna valor encontrado (string) ou ""
   'usa a constante Ini_File e
   'a API (Windows 3.1 Kernel) GetPrivateProfileString
   On Error GoTo Erro_LeMeuINI
   Dim VL_Return As String, VL_FileName As String
   Dim VL_SizeHandle As Integer, VL_Valid As Integer
   Dim Va_Msg As String
   Const CL_Default = "" 'retorno no caso de não encontrar
   VL_Return = Space$(VL_Size) 'string  a retornar
   VL_SizeHandle = Len(VL_Return) 'tamanho da string de retorno
   VL_FileName = Ini_File 'arquivo no formato INI
   VL_Valid = GetPrivateProfileString(VL_Sec, VL_Key, CL_Default, _
                VL_Return , VL_SizeHandle, VL_FileName)
   FU_Le_MeuIni = Left$(VL_Return, VL_Valid)
   Exit Function
Erro_LeMeuINI:
   FU_Le_MeuIni = CL_Default
   Resume Next
End Function
```

A função usada como **exemplo** é do VB3, podendo ser usada em VB4 se sua aplicação for em 16 bits. Mas, o conceito de criar funções "traduzidas" ou "facilitadas" de API é aplicável a qualquer versão do Visual Basic.

Por Charles A. Müller

33 - VB4 - Criando senhas para banco de dados

O **Jet Engine 3** (exclusivo32 bits) inclui um novo sistema de segurança baseado em senhas de BD mais complexas e mais seguras que o antigo modelo de grupos. Este sistema disponibiliza uma senha para abertura da base de dados . Este sistema é mais simples de ser utilizado mas é facilmente comprometido, pois, todos os usuários possuem a mesma senha. Entretanto, você poderá usar tanto o recurso de *DB Password* (senha de BD) como o de *workgroup* (grupos), **ao mesmo tempo** (isto é, que dará mais segurança).

Manipule uma DB Password no VB, usando o novo método *NewPassword* (*database object*), com códigos como este:

```
Dim wrk As Workspace
Dim db As Database
```

```
Set wrk = DBEngine. Workspace(0)
Set db = wrk.OpenDatabase("MYDB.MDB",true)
'note que a base deve ser aberta como exclusiva
'alterando a senha atual (em branco) para "NewPass"
db.NewPassword "","NewPass
```

*Por Paul Litwin**

34 - VB4 - Abrindo bases de dados com senha

Na *dica anterior* mostrei a definição de senhas para bancos *Jet 3* (32 bits). Para abrir o banco é necessário passar a senha no parâmetro *Connect*. No exemplo abaixo, a senha é "bobo".

```
Dim wrk As Workspace
Dim db As Database
Set wrk = DBEngine. Workspace(0)
Set db = wrk.OpenDatabase("MYDB.MDB", false, false, ";PWD=bobo")
```

O parâmetro *Connect* (4º parâmetro) é *case sensitive* (diferencia *A* de *a*) e - ao contrário do que diz a documentação do VB - os parâmetros *exclusive* e *read-only* (2º e 3º parâmetros) devem ser **falsos**.

*Por Paul Litwin**

35 - VB4 - Posicionando uma Common Dialog

Ficou triste ao ler a documentação do VB, que dizia "Note: you cannot specify where a common dialog is displayed" (você não poderá especificar onde é mostrada uma common dialog)? Então tente isto:

Inicie um novo form (que será usado apenas para isto) em vez de chamar a abertura do diálogo diretamente do form principal.

(FrmDummy_OpenSaveAs.Hide)

Defina as propriedades *Left* e *Top* conforme desejar e inicie a *common dialog* deste form. No Windows 95 (VB 4-32 bits) , a common dialog irá aparecer na posição do form que a chamou. Como o form *hide* (oculto), isto é imperceptível para o usuário.

*Por Reinhard Salchner**

36 - VB3/VB4 - Economize memória com uma picture box

Mudar a propriedade ***AutoRedraw*** para true consiste em redesenhar forms rapidamente e desperdiçar alguma memória. Se seu form é redimensionável, o desperdício pode ser bem maior, pois, o bitmap persistente criado pelo *AutoRedraw* é tão grande quanto as dimensões máximas do form para revelar a saída oculta, quando o usuário maximiza ou minimiza a janela. Se o gráfico a ser redimensionado (mantido)

for pequeno em relação ao form, você economizará memória se utilizar uma ***picture box*** com *AutoRedraw* = *true* e *BorderStyle* = 0, enquanto o *AutoRedraw* do form será desativado (*false*).
*Por Francesco Balena**

37 - VB3/VB4 - Lembra-se do SWAP?

Fiquei surpreso quando notei que no Visual Basic, o comando SWAP do Qbasic não havia sido implementado. Na rotina abaixo, que usei para ordenar um arquivo, o SWAP é simulado com strings, mas funciona com outros tipos de dado.

```
Private Sub Form_Load( )
        Dim a,b As String * 4
        Dim c As String * 4 ' variável para alternação (Swap)
        a = "João"
        b = "Francisco"
        Debug.Print "Antes do swap: " & a & " " & b
        c = a
        a = b
        b = c
        Debug.Print "Após o swap: " & a & " " & b
End Sub
```

*Por David Ferber**

38 - VB3/VB4 - Uma história de três beeps

Seus programas não estão executando instruções em VB4 como executavam em VB3? Tente isto , em Qbasic, VB3 e VB4.
BEEP: BEEP: BEEP
Ao depurar com passo (F8), este mui complexo código, você irá ouvir três *Beeps*, exceto no VB4. **No VB4, palavras reservadas seguidas de dois pontos (:) são consideradas *labels* (rótulos de desvio).**
Assim funciona:

```
Beep
Beep
Beep
```

E você ouvirá os tão esperados três beeps.
*Por David Ferber**

39 - VB3/VB4 - Conversão de Nulos

Em consultas a bancos de dados, o retorno de uma variável, quando nula, poderá não ser 0 (numérico) ou "" (string). Geralmente se resolve assim:

```
If Not IsNull(myrecordset.myfield) Then
        myvar = myrecordset.myfield
Else
        myvar = ""
        'myvar = 0, no caso de numéricos
End If
```
Uma forma mais simples é-
```
myvar = "" & myrecordset.myfield
```
Ou
```
myvar = val(0 & myrecordset.myfield) ' para numéricos
```
Por Garold Minkin aperf. por Charles A. Müller*

40 - VB4 - Determinando a classe de qualquer objeto

No VB4, o comando *TypeOf* trabalha com qualquer objeto válido. Exemplo:
```
'Esta rotina imprime informações específicas de objetos
Public Sub PrintObjectInfo (YourObject As Object)
        If TypeOf YourObject Is CDesk then
                Print "Object Type: Mesa"
                Print "Número de pernas: " & YourObject.NumberOfLegs
        ElseIf TypeOf YourObject  Is CHouse Then
                Print "Object Type: Casa"
                Print "Número de portas: " & YourObject.NumberOfDoors
        End If
        'impressão das propriedades de mesmo nome
        Print "Data de Venda: " & YourObject.Date
        Print "Preço de Venda: " & YourObject.Price
        '...
End Sub
```
Por Hassan Davis, MicroHelp Inc*

41 - VB4 - Identificando um controle genérico

Quando uma rotina pode trabalhar com muitos tipos de controles diferentes, a função *TypeOF* pode detectar o tipo de controle em tempo de execução:
```
Function MyFunc (ctl as Control)
        If TypeOf ctl Is TextBox Then
                '...
        ElseIf TypeOf ctl Is CommandButton Then
                '...
        '...
        End If
End Function
```

Este código funciona em VB3 e VB4. A diferença é que no VB4, além de controles e forms, qualquer objeto válido pode ser identificado. O **VB4** adiciona ainda, a função *TypeName* que indica (numa *string*) o nome da classe do objeto:

```
Function MyFunc (ctl as Control)
        Dim sClassType As String
        'typeName é novidade do VB4
        sClassType = TypeName(ctl)
        Select Case sClassType
                Case "TextBox"
                        '...
                Case "CommandButton"
                        '...
                'case ...
        End Select
End Function
```

Os nomes das classes de controle, no ambiente do VB, aparecem na *Properties Window* (janela de propriedades, ao lado do nome do controle).
*Por Senthil Shanmugham**

42 - VB3/VB4 - Removendo o move

Em alguns casos, é interessante impedir o usuário de mover um form. No VB isto pode ser implementado com APIs:

```
Declare Function GetMenu% Lib "User" (ByVal hWnd%)
Declare Function RemoveMenu% Lib "User" (ByVal hWnd%, ByVal nPosition%, ByVal
wFlags%)
'...
Dim Res%
Res = RemoveMenu(GetMenu(Form.hWnd), SC_MOVE, MF_BYPOSITION)
```

*Por Phil Parsons**

43 VB4 - Otimizando consultas no Jet 3

Se você precisa analisar a performance de uma *query* (consulta) no *Jet Engine 3.0* (banco .MDB), através de um plano de execução de consultas, você deve adicionar esta chave de *Registry* e executá-la no RegEdit.

\\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\3.0\Engines\Debug

Para a nova chave Debug, adicione o nome JETSHOWPLAN (tudo maiúsculo) e valor ON. O Jet irá gerar um arquivo chamado SHOWPLAN.OUT, que irá mostrar planos de execução de *queries* associados com sua aplicação. Como estes arquivos podem se tornar muito grandes rapidamente, não se esqueça de alterar o valor para OFF ao terminar.

Queries e bases de dados bem definidas irão gerar planos que, indicarão o uso de índices e/ou a tecnologia *Rushmore*. Bases e consultas mal definidas exibem apenas uma leitura de tabela.

*Por Rob Karatzas**

44 - VB3/VB4 - Piscar ou não piscar

Geralmente, ao criar uma ajuda de barra de situação (*status bar help*), você irá adicionar código no evento *MouseMove* de controles e forms. A barra de status poderá ser uma picture box com um label, um controle statusbar (VB4-32 bits) ou - como mais usado - um painel 3D. O problema é que o mouse se movimenta várias vezes no mesmo controle, fazendo a barra de status "piscar". Para resolver este problema, basta verificar se a frase atual é diferente da frase nova. Criamos ainda uma função que controla a barra de status.

```
Global Const CG_EXPLICAPADRAO = "Pressione F1 para obter ajuda."
'...
Sub SU_Explica (oque As String)
    'rotina em VB3, usa um painel 3D
          oque = Trim(oque)
   If oque = "" Then oque = CG_EXPLICAPADRAO
   If Len(oque) > 1 And Right$(oque, 1) <> "." Then
      oque = oque & "." 'acrescenta ponto final
   End If
         'muda a inicial para maiúscula
   If Len(oque) > 1 Then
      oque = UCase(Left(oque, 1)) & Right(oque, Len(oque) - 1)
   Else
      oque = UCase(oque)
   End If
   'altera a barra se o novo conteúdo for diferente
   If MainForm.PA_Status.Caption <> oque Then
      MainForm.PA_Status.FloodShowPct = False
      MainForm.PA_Status.FloodType = 0
      MainForm.PA_Status.Caption = oque
   End If
End Sub
```

A função usa uma constante (CG_EXPLICAPADRAO) que, contém uma frase genérica, para quando não houver o que explicar sobre um form ou objeto.

Para chamar a função:

```
Sub CmdOK_MouseMove(Button As Integer, Shift As Integer, X As Single, _
         Y As Single)
         SU_Explica "Grava as alterações no banco de dados."
End Sub
```

Pode ser utilizado também o evento *GotFocus* (para resposta ao teclado). Neste caso, o *LostFocus* deverá limpar a barra de status (como a frase genérica).

```
Sub txtNome_LostFocus ( )
         SU_Emplica ""
End Sub
```

Por Dave Robins, aperf. por Charles A. Müller*

As três próximas dicas se referem ao que mostrar para os usuários em processos demorados:

45 - VB3/VB4 - Travou tudo?

Em alguns processos demorados, como consultas a bancos de dados, o usuário tem a impressão de que o sistema "travou". No Windows 3.1x, esta sensação é maior (pois o controle de tarefas é mais rudimentar). Para minimizar o problema, estes *loops* (laços de repetição de código) demorados devem conter uma instrução ***DoEvents***. Para não assustar o usuário, é alterada a propriedade ***MousePointer*** do Form para ampulheta (*hourglass*) ou aparece uma mensagem (ou percentuais) na barra de status.
Para mudar o ponteiro do mouse:

```
'antes
Me.MousePointer = 11 'hourglass (ampulheta)
ExecutarProcessoDemorado
'depois
Me.MousePointer = 0 'padrão
```

Neste caso, se o sistema operacional for o Windows 95 (ou outro Win32), o usuário poderá definir um ícone animado nas suas configurações.
Outra forma é utilizar um *3D Panel* como **barra de progresso percentual**, usando as propriedades *FloodPercent*, *FloodShowPct* e *FloodType* (detalhes no help do VB).
Estes são os recursos mais comuns. Mas, se a barra de status já estiver sendo utilizada como contador percentual, onde exibirei uma mensagem de "aguarde processando..."?
E se eu desejar algo com maior destaque que um simples ponteiro de mouse?
Não é possível utilizar um form para isto ou uma caixa de mensagem, pois, eles esperariam uma ação do usuário - o que interromperia o processamento.
A solução é incluir, no MDI Form (form principal), uma **"faixa de aguarde"**. Assim:

1) No MDI Form de sua aplicação (aqui chamado de F00), insira uma *picture box*, que será chamada *PI_Aguarde*. Esta picture box será como uma barra de ferramentas. A propriedade *Align* deverá ser 1 (*Align Top*). Esta é a "faixa de aguarde". Atribua *false* para a propriedade *Visible*.
2) Na PI_Aguarde, insira um rótulo (*label*), chamado *LB_MsgAguarde*. Use um tamanho e formato de fonte que dê bastante destaque ao texto.
3) Você poderá inserir ainda, ao lado do label, uma outra picture (pequena e para enfeite), contendo um desenho que remeta à idéia de espera. Este desenho, poderá ser um ícone de um semáforo.
4) Em um módulo (.BAS), insira a rotina *SU_Aguarde*, para manipular a faixa.

Eis o código da rotina:

```
Sub SU_Aguarde (VA_Liga As Integer, VA_Msg As String)
   'recebe VA_Liga (true/false)
   'mostra a picture de aguarde com VA_Msg ou padrão
         If VA_Liga Then
      F00.PI_Aguarde.Visible = True
      screen.MousePointer = 11 'hourglass
      VA_Msg = Trim$(VA_Msg)
      If Len(VA_Msg) = 0 Then
                                 'mensagem padrão
         VA_Msg = "Por favor, aguarde: processando..."
```

```
    End If
    F00.LB_MsgAguarde.Caption = VA_Msg
  Else 'desliga
                    F00.LB_MsgAguarde.Caption = ""
                    F00.PI_Aguarde.Visible = False
    screen.MousePointer = 0 'default
  End If
End Sub
```

Esta rotina pública passa mensagens para a faixa, que ficará ativa durante o processo demorado. O primeiro parâmetro (true/false) liga ou desliga a barra. O segundo passa uma frase. No caso de frase vazia (""), é usada uma frase padrão.
Para chamar a rotina:

```
'antes
SU_Aguarde True, "Por favor, aguarde: consultando tabela de Clientes..."
ExecutarConsultaGrid_Cliente
'depois
SU_Aguarde False, ""
```

Por Charles A. Müller

46 - VB3/VB4 - Painel de Percentual

Na *dica anterior*, citei a **barra de progresso percentual** como uma forma de mostrar ao usuário como está um processo demorado (assim ele não pensará que o programa "travou"). Para mostrar um percentual, é preciso conhecer o tempo (ou tamanho) total da operação e a que ponto se está em dado momento de um *loop* (laço de repetição). Num programa de instalação, por exemplo, se conhece o tamanho total dos arquivos (ou quantidade de arquivos) a serem instalados e qual o arquivo atual (no loop). Com isto, o usuário vê X% da instalação completa.
Para usar um *3D Panel* como barra de percentual, siga estes passos:

1) Insira um 3D Panel, com nome *PA_Status*, no MDIForm (aqui chamado de F00). Atribua *Align = Alig Botton*.
2) Insira em um módulo (.BAS) a rotina *SU_BarraPerc*.

```
Sub SU_BarraPerc (Perc As Integer, Acum As Integer)
        'recebe perc, um número de 0 a 100
  '100 = "desliga" a barra
  'Acum = boolean, acumula o anterior ou não (true/false)

  Static VA_Vez
  Static VA_SaveCor As Long

  If Acum Then
    Perc = Perc + F00.PA_Status.FloodPercent
  End If
  If Perc > 100 Or Perc < -1 Then
    MsgBox "Perc deve estar entre -1 e 100", 16, "Erro de parâmetro _
                                          em SU_BarraPerc"
    Exit Sub
  End If
```

```
   If IsEmpty(VA_Vez) Or VA_Vez = 1 Then
      'liga barra - altera o painel
                           F00.PA_Status.Caption = ""
                           F00.PA_Status.FloodShowPct = True
      VA_SaveCor = F00.PA_Status.ForeColor
      F00.PA_Status.ForeColor = RGB(0, 0, 0)'preto
                           F00.PA_Status.BevelOuter = 2 'raised
      F00.PA_Status.BevelWidth = 3
      F00.PA_Status.BorderWidth = 1
      F00.PA_Status.FloodType = 1 'left to right
      F00.PA_Status.FontSize = 9.75
   End If
   If Perc < 100 Then
      If Perc > 48 Then
         F00.PA_Status.ForeColor = RGB(255, 255, 255)'branco
      End If
      'mostra perc
      F00.PA_Status.FloodPercent = Perc
      VA_Vez = 2 'ou mais
   Else
      'desliga barra - reestrutura painel
                           F00.PA_Status.BevelOuter = 1 'inset
      F00.PA_Status.BevelWidth = 1
      F00.PA_Status.BorderWidth = 3
      F00.PA_Status.FloodType = 0 'none
      F00.PA_Status.FontSize = 8.25
      F00.PA_Status.ForeColor = VA_SaveCor
      F00.PA_Status.FloodShowPct = False
                           VA_Vez = 1
   End If
End Sub
```

Para chamar a rotina, basta passar o valor atual do percentual. O segundo parâmetro, indicará se o percentual anterior será acumulado com este. No exemplo abaixo, a barra é preenchida de 10% em 10%.

```
'teste da barra de percentual
 Dim i As Integer
 For i = 1 To 10
    SU_BarraPerc (i * 10), False
                MsgBox "Clique em OK para continuar"
 Next i
 SU_BarraPerc (100), False 'desliga a barra
```

Por Charles A. Müller

47 - VB3/VB4 - Painel de Percentual com SQL Count

Complementando a *dica anterior*: Em uma operação de consulta a um banco de dados (típica de desenvolvimento comercial), deveremos conhecer o tamanho do retorno da consulta. O número de linhas que irá retornar é calculado por um *Select Count*

(instrução **SQL** para contador) igual ao *Select* que, posteriormente, será usado para a consulta. O Count é uma operação rápida, principalmente em bancos Client Server (onde o cálculo é executado no servidor). O retorno do Select Count é um número, contendo o total de linhas que seria trazido pela consulta. Com o Count, poderão ser impedidas consultas longas demais, por exemplo.
Para o percentual, já temos o total. O "registro corrente" é obtido dentro do loop. No exemplo abaixo, carregamos um *Grid* simples com dados de uma tabela. Utilizamos as rotinas SU_Aguarde e SU_BarraPerc (explicadas nas dicas anteriores).

```
Sub SU_CarregarGrid ()
   Dim VA_Cmd As String
   Dim dynatemp As dynaset
   Dim dynacont As dynaset
   Dim VA_Cont, VA_Curr
   Dim VA_SevErro

   On Error GoTo Erro_Carregar_Grid
   SU_Aguarde True, "Carregando tabela de cidades..."
   'rotina acima explicada na DICA ANTERIOR
   '... limpar o Grid
   '... formatar TB_Cidade.text

   'query
   VA_Cmd = "Select * From CIDADE"
   If Len(TB_Cidade.Text) > 0 Then
      VA_Cmd = VA_Cmd + " Where CIDADE.Nome >= '" & (TB_Cidade.Text) & "' "
      VA_Cmd = VA_Cmd + "And CIDADE.Nome <= '" & (TB_Cidade.Text) & Chr(255)
                        & "' "
   End If
   Set dynatemp = db.CreateDynaset(VA_Cmd, VGl_SQLop)

   'query do contador
   VA_Cmd = "Select Count(*) From CIDADE"
   If Len(TB_Cidade.Text) > 0 Then
      VA_Cmd = VA_Cmd + " Where CIDADE.Nome >= '" & (TB_Cidade.Text) & "' "
      VA_Cmd = VA_Cmd + "And CIDADE.Nome <= '" & (TB_Cidade.Text) & Chr(255)
                        & "' "
   End If
   Set dynacont = db.CreateDynaset(VA_Cmd, VGl_SQLop)
   If Not dynacont.EOF Then
      VA_Cont = dynacont(0)
   Else
      VA_Cont = 0
   End If
   If VA_Cont = 0 Then
      MsgBox "Nenhum registro de cidade encontrado."
      SU_Aguarde False, ""
      Gr_Grid.Row = 1
      '... marcar outra linha do grid
      Exit Sub
   End If
   'carga do grid
```

```
   Gr_Grid.Rows = VA_Cont + 1
   VA_Curr = 1
   Do While Not dynatemp.EOF
      SU_BarraPerc CInt(VA_Curr * 100 / VA_Cont), False
      'rotina explicada na DICA ANTERIOR
                        Gr_Grid.Row = VA_Curr
      Gr_Grid.Col = 0
      Gr_Grid.Text = dynatemp("CodCidade")
      Gr_Grid.Col = 1
      Gr_Grid.Text = dynatemp("NomeCidade")
      Gr_Grid.Col = 2
      Gr_Grid.Text = dynatemp("UF")
      VA_Curr = VA_Curr + 1
      dynatemp.MoveNext
   Loop
    'desliga a barra de percentual
                SU_BarraPerc 100, false
                '... demais lógicas
                '... (tratamento de erro e formatações)
End Sub
```

Como esta rotina de carga de grid é enorme (e no VB4, o *DBGrid* faz isto sozinho), o código acima apenas mostra a formação do Select Count e a chamada a SU_BarraPerc.
Por Charles A. Müller

48 - VB3 - Mantendo constantes

Melhore o uso do arquivo CONSTANT.TXT. Para um novo projeto, copie o arquivo CONSTANT.TXT para MYCONST.TXT (para o diretório do seu projeto). Inclua MYCONST.TXT no seu projeto (menu *File → Add File*). Substitua (menu *Edit → Replace* ou CTRL + R) todas as expressões ***Global*** por ***' Global*** neste arquivo.
Quando for necessária uma nova constante, basta verificar se a mesma já foi definida pela Microsoft e remover o a **'** do comentário (reverter a substituição).
*Por Stan Mlynek**

49 - VB3/VB4 - Inconsistência no caminho da aplicação (app.path)

Esteja atento quando usar a propriedade *path* (caminho) do objeto Application (*App*, aplicação). Se seu executável está rodando na raiz de um drive, App.Path retornará o nome (letra:) na unidade e uma barra (algo como C:\). Apareceu quando o executável está em um subdiretório, a barra final não é acrescentada (C:\SUBDIR). Para testar e acrescentar a barra, use o código abaixo que, retornará C:\SUBDIR\.

```
MyPath = App.Path
If Not Right(MyPath, 1) = Chr(92) then
        'chr 92 = "\"
```

```
        MyPath = MyPath & Chr(92)
End If
```
*Por Clint Walker**

50 - VB3/VB4 - Bloqueando funções Copiar e Colar em caixas de texto

As funções Copiar (CTRL+C) e Colar (CTRL+V) estão sempre disponíveis para text boxes, mas e se você não desejar que estas funções funcionem? Você deve supor que o evento *KeyDown* consegue detectar CTR+C e CTRL+V, mas não detecta. No evento *KeyPress*, estas teclas podem ser capturadas:

```
Sub Text1_KeyPress (KeyAscii As Integer)
   If KeyAscii = 3 Or KeyAscii = 22 Then KeyAscii = 0
   'CTRL+C = 3 e CTRL+V = 22, valores não constantes na tabela ANSI,
         'geram estas combinações. Recurso não documentado
End Sub
```
*Por Pedro Velazquez Dávila**

51 - VB3/VB4 - Digitação em Grid

O controle *Grid* é uma tabela de exibição de dados, que não permite a digitação direta de valores em suas células (não estou falando do *DBGrid*). É possível "simular" a aceitação de teclas através de código. Basta um clique de mouse na célula e digitar. O programador deve ter incluído as rotinas abaixo nos eventos *KeyDown* e *KeyPress*.

```
Sub Grid1_KeyDown (KeyCode As Integer, Shift As Integer)
   Select Case KeyCode
    Case &H8 'BACKSPACE
      If Len(Grid1.Text) > 0 Then
         Grid1.Text = Left(Grid1.Text, (Len(Grid1.Text) - 1))
      End If
    Case &H2E 'DEL
      Grid1.Text = ""
   End Select
End Sub
Sub Grid1_KeyPress (keyascii As Integer)
   Select Case keyascii
      Case Is <> 8, 9, 10, 13'não imprimíveis
         Grid1.Text = Grid1.Text & Chr(keyascii)
   End Select
End Sub
```
No evento *KeyPress* todos os caracteres imprimíveis são acrescentados ao texto da célula ativa do *Grid*. O evento *KeyDown* apaga o último caracter com BACKSPACE ou o texto inteiro com a tecla DEL.
Por Charles A. Müller.

52 - VB3/VB4 - O Caracter ENTER

Ele nunca aparece, mas existe. No Word é representado por um ¶, nas caixas de texto do VB aparece um ψ (caracter não imprimível). Mas como aceitar e gerar este caracter?

1) No evento *KeyPress*, o caracter digitado (parâmetro *KeyAscii*) vale 10 ou 13 (os dois valores do ENTER na tabela de caracteres ANSI).
2) Em quaisquer consistências (como *TextBox_Change* ou análise de variáveis *string*) a função Asc retornará 10 ou 13.

```
Texto = Left(Texto, (Len(Texto) - 1))
If Asc(texto) = 10 or Asc(texto) = 13 then msgbox "Foi digitado um ENTER"
```

3) Numa *textbox*, é preciso gerar (via função *chr*) o caracter 10 ***mais*** o caracter 13.

```
text1.Text = "linha 1"
text1.Text = text1.Text & Chr$(13) & Chr$(10) & "linha 2"
text1.Text = text1.Text & Chr$(13) & Chr$(10)
text1.Text = text1.Text & Chr$(13) & Chr$(10) & "linha 4"
```

4) Numa *MsgBox*, basta gerar o caracter 13.

```
Dim vmsg As String
vmsg = "linha 1"
vmsg = vmsg & Chr$(13) & "linha 2"
vmsg = vmsg & Chr$(13)
vmsg = vmsg & Chr$(13) & "linha 4"
MsgBox vmsg, 0, "texto 2"
```

Por Charles A. Müller.

53 - VB3/VB4 - Limpando Combos Read-Only

Numa *ComboBox* com a propriedade *Style* = 2 (*dropdown list*), a propriedade *Text* é somente-para-leitura. Isto impede limpeza e troca de conteúdo por esta propriedade, em construções como estas:

```
Combo1.text = "" 'ou
Combo1.text = "novo conteúdo"
```

A solução é limpar a *combo* com o método *clear* e adicionar o valor novo.

```
Combo1.Clear
Combo1.AddItem "novo conteúdo "
```

Por Charles A. Müller.

54 - VB3/VB4 - Brancos no controle Masked Edit Box

O controle *MS Masked Edit* apenas aceita entrada de dados dentro da máscara formatada (*mask*). Isto impede o programador de limpar a text do controle diretamente (masked1.text = ""), pois, o caracter espaço (ou nulo) pode não se encaixar no formato da máscara. Por exemplo, algumas possuem o formato # (aceitam somente números). Logo, o "" não seria aceito. Este problema é resolvido por este código:

```
vTemp = masked1.mask
masked1.mask = ""
masked1.text = ""
masked.mask = vTemp
```

Removendo a máscara é possível limpar o texto. Depois, basta devolver a máscara original ao controle. Uso isto no evento Data1_*ValidationError* quando adiciono um novo registro.
*Por Scott Wallace**

55 - VB3/VB4 - Forçando caracteres maiúsculos

Para facilitar a digitação de maiúsculos, independente do pressionamento de CAPS LOCK, converta cada caracter no evento *KeyPress*.

```
Private Sub Form_KeyPress (KeyAscii as Integer)
        KeyAscii = Asc(UCase(Chr(KeyAscii)))
End Sub
```

Para que esta rotina funcione para todos os campos do form, altere a propriedade *KeyPreview* do mesmo para *true*.
*Balamurali Balaji**

56 - VB3/VB4 - Pinte meu mundo ... nas cores padrão!

Apenas após executar o último "*make EXE*", troque seu esquema de cores e veja quantos fundos de cores você possui fora do padrão (**escolhido pelo usuário final, via Painel de Controle**). Infelizmente, muitos controles customizados (VBX/OCX) pecam neste detalhe importante.
Através do Painel de Controle, tente o esquema "deserto" no Windows 95, ou "verão" no Windows 3.1x, ou ainda, crie um outro esquema horroroso. Isto o ajudará a testar suas aplicações de cores.
O VB4 dispõe de 24 cores de sistema como constantes. No VB Help, procure por "*Color Constants*" ou "*VBTranslateColor*". Copie o valor hexadecimal para a propriedade de cor de seu objeto se ele foi erroneamente redefinido (fora do padrão). Também é possível copiar os valores de controles que estejam corretos, mas cuidado, a face do botão por exemplo, pode não ser cinza.
*Por Clint Walker**
*Nota do VBPJ**: Na nova paleta de cores do VB4, há um botão ***Default***, que altera as cores do objeto selecionado para o padrão do Windows. Verifique se a propriedade *Appearance* é *3D* para obter melhores resultados.
Notas do Fórum Access (Charles A. Müller): 1) É uma regrinha antiga. **Nunca** mude as cores que, o usuário, que é o **cliente**, define externamente (no Windows), **a menos que isto traga utilidade**. Nestes casos, esteja atento para o significado das cores e sua harmonia (afinal, penteadeiras de camarim na tela não são desejáveis). 2) No VB3, existem 19 cores de sistema nas constantes de CONSTANT.TXT. 3) Observamos (no CCE e VBA) que o VB5 terá, em sua paleta de cores, uma lista de alteração automática para as cores padrão (além do botão *default*).

57 - VB3 - Desmarcar todos os itens de uma lista

Uma forma rápida de retirar qualquer seleção de uma *listbox* é:
list1.selected (-1) = False
Isto não funciona em VB4.
*Por John Müller**

58 - VB4 - Ordenando Colunas da ListView

Dê ao seu controle *ListView* (32 bits) a funcionalidade de ordenação do Windows 95 Explorer. Este código ordena a lista por qualquer coluna. Se a lista já estiver ordenada por esta coluna, a ordem será invertida.

```
Private Sub ListView1_ColumnClick _
 (ByBal ColumnHeader As ColumnHeader)
        Whith ListView1
                If (ColumnHeader.Index -1) = .SortKey Then
                        .SortOrder = (.SortOrder + 1) Mod 2
                Else
                        .Sorted = False
                        .SortOrder = 0
                        .SortKey = ColumnHeader - 1
                        .Sorted = True
                End IF
        End With
End Sub
```

*Por Joe Tuttle**

59 - VB4 - Problemas com o Print

O código abaixo funciona em VB3:

```
Cls
Print Spc(10); "Informe seu nome:";
currentX = 0
currentY = currentY +1
Print Spc(10); "Informe seu nome:";
```

Este código falha em VB4. Retirando o último ponto-e-vírgula do primeiro print:

```
Print Spc(10); "Informe seu nome:"
```

Acrescente Debug. antes de Print para testar:

```
Cls
Debug.Print Spc(10); "Informe seu nome:"
currentX = 0
currentY = currentY +1
Debug.Print Spc(10); "Informe seu nome:";
```

Ou mude o comando para:
Print Space(10); "Informe seu nome:";
*Por David Ferber**

60 - VB4 - Use o Code Profiler para depuração (debug)

Algumas vezes, um erro de execução se manifesta apenas após a criação de um EXE e não em tempo de *debug*. O add-in *Code Profiler* poderá ajudá-lo.

1) Faça uma cópia do seu fonte.
2) Selecione o add-in Code Profiler.
3) Selecione o(s) arquivo(s) de código a serem analisados.
4) Selecione a opção *Line Hit Count*.
5) Selecione o botão *Add Profiler Code*.
6) Compile (*make EXE*) o programa (MYAPP.EXE).
7) Execute o seu código com erro.
8) Volte ao Code Profiler e selecione *View Results* no menu *File*.

Veja a última linha que foi executada ao ocorrer o erro. Você terá que executar seu código em modo debug enquanto olha os resultados do Code Profiler.
*Por Rich Spencer**

61 - VB3/VB4 - Onde está o Beep?

Este código elimina o beep quando se tecla ENTER ou TAB em uma text box que atingiu seu número máximo de caracteres.

```
Sub Form_KeyPress (keyascii as integer)
        If KeyAscii = 13 or KeyAscii = 9 Then
                KeyAscii = 0
        End If
End Sub
```

*Por Lonnie Brioadnax, Michael Ottomanelli e Preston Werntz**

62 - VB3/VB4 - TAB automático para o próximo campo

Esta dica é útil para desenvolvimento de aplicações VB com a forma de edição de terminais 3270 (IBM Mainframe). Quando o usuário termina de preencher um campo em um terminal 3270, o foco é imediatamente transmitido para o próximo campo.

```
Sub Text1.KeyUp (keycode as integer, shift as integer)
        If keycode > 47 and keycode < 123 then
                If Len(Me.ActiveControl.Text) = (Me.ActiveControl.MaxLenght) then
                        Sendkeys "{TAB}"
                End If
        End If
```

End Sub
*Por Lonnie Brioadnax, Michael Ottomanelli e Preston Werntz**

63 - VB3/VB4 - Simplificando a condição de um IF

Quando você escreve um comando IF (Se) assim:

```
If Category = "CM" or Category = "M2" or Category = "P1" or Category = "ZZ" then
        ProcesseEmpregado
End If
```

Poderia simplificar para:

```
Dim ValidValues as string
ValidValues = "CM M2 P1 ZZ"
If (InStr(1, ValidValues, Category)) > 0 then
        ProcesseEmpregado
End If
```

Isto torna o código mais rápido e mais fácil de ser entendido. Note que separei os valores com " " para não aparecerem strings como "CMM2P1ZZ"; você poderá utilizar espaços ou outros separadores como vírgulas, ponto-e-vírgulas etc.
*Por Jaspreet Singh**
Notas de Redação (Charles A. Müller):

1) Este teste ainda aceitará "C", " P", "2 P" e outras expressões inválidas, por conterem o separador ou terem tamanho inválido. Para corrigir esta falha, basta usar espaço (apenas) como separador e testar o tamanho da categoria.

```
Category = trim(Category)
If (InStr(1, ValidValues, Category)) > 0 and Len(Category) = 2 Then
'...
```

2) Este truque pode ser usado para várias validações, como ValidValues = "abcdefghijlmnopqrstuvxz".
3) Outra forma, ainda mais fácil, é criar um pequeno *Select Case*, separando os valores válidos por vírgulas:

```
Select Case Category
  Case "CM",  "M2", "P1", "ZZ"
        ProcesseEmpregado
End Select
```

64 - VB3/VB4 - Eliminando o IF quando possível

Se você atribui *true* ou *false* para uma variável (ou propriedade), após testar certas condições, poderia fazê-lo sem o IF. Veja:

```
If (age > 18 and sex = "M") and (NecessitaSeContigente = true ) Then ServicoMilitar =
true
```

Pode substituir por:

```
ServicoMilitar = (age > 18 and sex = "M") and (NecessitaSeContigente)
```

Outro exemplo:

IF (age > 25 and Category = "M1") or (age > 35 and Category = "C1") or _
(Age > 45 and Category = "P1") then ExecuteDemissao
Poderia ser:
Dim condicao as Integer 'boolean
condição = (age > 25 and Category = "M1") or (age > 35 and Category = "C1")_ or (Age > 45 and Category = "P1")
If condicao Then ExecuteDemissao
*Por Jaspreet Singh**

65 - VB4 - Forms redimensionáveis sem barra de título

Se você alterar as propriedades (de um form) caption = "" e controlbox = false, uma borderstyle = 3 (fixed) irá ser mostrada. Diferente da borderstyle = 0 (none), as propriedades 3D (VB4) são mantidas. Utilizando borderstyle = 5 (sizable toolwindows, no VB4), você terá um form redimensionável.
É possível (VB3/VB4) alternar o conteúdo da Caption, limpando-a quando conveniente. E não se esqueça de acrescentar um botão de Fechar (unload) no seu form!
*Por Clint Walker**

66 - VB3/VB4 - Adicionando segurança a uma base de dados Jet

Para dar segurança a uma base de dados *Jet* (.MDB), versão 2.5 (Access 2/VB3/VB4-16 bit) ou versão 3.0 (Access 7/VB4-32 bit), siga estes passos:
1) Use o *Access Workgroup Administrator* para criar um novo grupo de trabalho, com uma não nula Workgroup ID.
2) Inicie o Access e altere a senha para o usuário default **Admin**.
3) Crie um novo usuário, adicione-o no grupo de Administração, com os privilégios de administrador. Remova a conta Admin do grupo de administradores.
4) Reinicie o Access, conectando-se como novo usuário, e altere a senha.
5) Execute o Access Security Wizard (para o Access 2, copie de www.microsoft.com/accdev).
6) Crie o(s) usuários e o(s) grupo(s) de usuário, definindo seus privilégios.
7) Não defina nenhuma permissão para o Admin.

*Por Paul Litwin**

67 - VB3/VB4 - Passe nothing aos forms com cautela

É uma boa idéia passar o valor *nothing* a variáveis de form para recuperar memória alocada pelo módulo. Executando este recurso para um form já carregado, entretanto, irá colocar o módulo em um estado confuso. Veja:
Form2.show
Set Form2 = nothing

```
Form2.show
MsbBox forms.count & " forms carregados"
Unload Form2
Unload Form2
```

A segunda linha do código tornou form2 nothing, mas o segundo use do form2.show irá mostrar uma segunda instância do form2. A Forms Collection irá conhecer as duas instâncias, mas apenas uma será descarregada (Unload Form2).
Para contornar este problema, em **VB4**, esteja certo que o form está descarregado. Não é possível executar Set Me = Nothing. Mas, com a estrutura For Each (não existente no VB3) é possível se conseguir o Nothing, no **evento Form_*Unload***.

```
Private Sub Form_Unload (Cancel As Integer)
        Dim Form As Form
        For Each Form In Forms
                If Form Is Me Then
                        Set Form = Nothing
                        Exit For
                End If
        Next Form
End Sub
```

*Por Willian Storage**

68 - VB3/VB4 - Prevenindo interação do usuário, via MousePointer e Enabled

Mudar a propriedade *MousePointer* do form não impede a ação do usuário, via mouse ou teclado, apenas altera o desenho do ponteiro.
Para impedir que o usuário interaja com o sistema em algumas operações, desenvolvi esta dica, aplicável a *MDI parent forms* (janelas principais de interface múltipla) e seus *MDI children forms* (janelas filhas). Em alguns processos demorados (como carga de banco de dados) mude a propriedade ***enabled*** de um MDI child para false, assim:

```
'antes
Me.Enabled = False
Me.MousePointer = 11 'hourglass (ampulheta)
ExecutarProcessoDemorado
'depois
Me.Enabled = True
Me.MousePointer = 0 'padrão
```

No caso de um MDI com muitos filhos ativos, crie uma ***Forms Collection*** e desative (enabled = false) cada form. Depois de desativá-los, use MDIForm.Hourglass = false.
*Por Al Gehrig Jr**

69 - VB4 - Depure simultaneamente o servidor OLE e a aplicação

O VB4 não apenas permite a criação de servidores OLE, mas, também permite depurar (*debug*) o servidor e a aplicação cliente ao mesmo tempo. Se você criar um servidor

OLE remoto, altere a propriedade *Instancing* para *Creatable SingleUse*. Isto tornará o debugging muito mais interessante.
Cada vez que a classe for chamada, a aplicação tentará criar outra instância do servidor. O servidor estará rodando em tempo de desenho, e o VB não iniciará outra cópia de si mesmo para carregar o servidor novamente. A solução, é, temporariamente, definir ***Instancing = Creatable MultiUse*** para uso nos testes. Não se esqueça de voltar para *Creatable SingleUse* antes de compilar o servidor OLE.
*Por L.J. Johnson**

70 - VB4 - Identificando uma unidade de CD em Rede

A API de 32 bits é bem mais rica que a de 16 bits. Entretanto, a função *GetDriveType* mostra os Drives CDs em Rede, apenas como DRIVE_REMOTE (de rede). Isto é uma verdade, mas não completa. Combine a chamada a GetDriveType com uma chamada a *GetVolumeInformation* para determinar se o drive é, ao mesmo tempo, de rede e CD.
A chamada indica o sistema de arquivos: FAT, NTFS, HPFS ou CDFS (CD File System).

```
Declare Function GetVolumeInformation  _
        Lib "Kernel32" _
        Alias "GetVolumeInformationA" _
        (ByVal lPRootPathName as String _
        ByVal lpVolumeNameBuffer As String _
        ByVal nVolumeNameSize As Long _
        ByVal lpVolumeSerialNumber As Long _
        ByVal lpMaximumComponentLenght As Long _
        ByVal lpFileSystemFlags As Long _
        ByVal lpFileSystemNameSize As Long) _
As Long
'...
pstrRootPath = "E:\"
pstrVolName = Space$(256)
pstrSystemType = Space$(32)
plngSysTypeSize = Clng(Len(pstr(SystemType))
plnVolNameSize = Clng(Len(pstrVolName))
plngRtn = GetVolumeInformation _
        (pstrRoothPath, pstrVolName, _
        plngVolNameSize, plngVolSerialNum,
        plngMaxFileNameLen, plngSysFlags, _
        pstrSystemType, plngSysTypeSize)
```

*Por L. J. Johnson**

71 - VB4 - Solução para bug no DBGrid

Há um sério bug (erro) em VB4, no controle *Databoud Grid* usado com forms modais. Por exemplo, crie três forms: form1, form2 e form3. Adicione um Command1 (botão) em

cada form. No evento click do botão em form1, chame o form2 como modal. No evento click do botão em form2 chame o form3 como modal. Adicione um DBGrid no form3. No evento click do botão em form3, use *unload* form3.
Execute o form1 e aperte nos referidos botões. No clique do terceiro botão, ocorre um erro de pilha (*stack error*) com o Visual Basic (tanto em 16 como em 32 bits). Rodando em Windows 3.1x, o sistema trava completamente.
Solução: não use DBGrid com forms modais. Se, entretanto, você precisar de um form modal, simule-o. Basta alterar a propriedade do form2 (o que chamou) para false. Você poderá criar uma property para fazer referência ao form que chamou.

```
With FormModal
        .propCaller = Me
        .Show
End With
```

Agora altere *Caller.Enabled* = *false* no evento *Load* do form "modal". Volte para true no evento *Unload*.
*Por Luis Miguel da Costa Pereira Ferreira**

72 - VB4 - Propriedade Count, de Control Array, não documentada

No VB4, cada control *array* (vetor de controles) é uma *collection* e possui uma propriedade *Count*. Isto não ocorre com o VB3. É possível, então, se criar um *loop* (laço de repetição) tendo o Count como valor máximo.
Esta característica não aparece nem nos manuais, nem no help do VB4. Talvez, pelo fato de que uma control array collection não possui todas as propriedades e métodos das demais collections. A propriedade Count e o método Item são suportados, enquanto os métodos Add e Remove não o são.
Este pequeno exemplo usa o Count para determinar qual elemento de um vetor de botões de opção foi selecionado.

```
Private Sub FindSelectedOption ( )
        Dim ij As Integer  As Integer
        For ij = 0 to Option1.Count - 1
                If Option(ij).Value Then
                        MsgBox str(ij),0, "Opção Selecionada"
                End If
        Next ij
End Sub
```

Esta rotina trabalha apenas com números contínuos. Se os elementos forem 0, 1, 3 e 4, ocorrerá um erro (run time error 340), ao se tentar fazer referência ao item 2.
*Por Craig Everett**

73 - VB4 - Determinando se um objeto foi definido (Set)

VB4 providencia uma série de novas capacidades de uso de objetos. Porém, um objeto deve ser "setado" (definido) antes de ser referenciado. A única forma de verificar se um objeto já foi definido é através do código de erro (91).

Por exemplo:

```
Public Function IsSomething (ob As Object) As Long
        Dim J as Long
        Err.Clear
        On Error Resume Next
        If TypeOf ob Is TextBox Then
                J = 1
        End If
        Select Case Err.Number
                Case 91
                'error 91 = object not set
                        IsSomethig = false
                Case 0
                        IsSomething = true
                Case Else
                        '... outro erro ocorreu
        End Select
        On Error GoTo 0
End Function
```

*Por Evan Dickinson**

74 - VB3/VB4 - Criando Inner Joins (SQL) numa base Access (Jet)

A palavra reservada In, da linguagem SQL (estrutura Inner Join) funciona em bases externas (ODBC), mas causa problemas em bases do Access (MDB). A sintaxe correta é:

```
SELECT Authors.*
FROM C:\VB\Biblio1.Authors
INNER JOIN C:\VB\Biblio2.MDB.Titles
INNER JOIN C:\VB\Biblio3.MDB.Publishers
INNER JOIN C:\VB\Biblio4.MDB.[Publisher Comments]
ON Publishers.PubID = [Publisher Comments].PubID
ON Titles.PubID = Publishers.PubID
ON Authors.Au_ID = Titles.Au_ID
```

Os comandos de uso do SQL (como CreateDynaset) devem estar em uma só linha (ou _, no VB4). Usamos como exemplo, a base BIBLIO.MDB dividida em 4 bases, uma com cada tabela.

Ao usar ODBC, trabalhe com a cláusula IN. Veja no Help do VB detalhes sobre o SQL.

Por Mark P. Atwood, texto revisado por Charles A. Müller*

75 - VB4 - O desafio de criar Add-ins

Escrever *add-ins* (recursos adicionais) para o VB4 pode ser desafiador, recompensador e melindroso. Se você não tomar cuidado, o VB poderá "estranhar" algumas coisas e

abortar. Podem aparecer várias mensagens, dependendo do sistema operacional. As mensagens são diversas, mas o resultado é o mesmo.
Por exemplo, no Windows 95, aparecem mensagens como "Este programa causou um erro e vai ser encerrado" ou "se o problema persistir, contate o fornecedor". No Windows 3.1x, podem ser causados GPFs.
Este erros ocorrem quando a IDE está sendo descarregada (*unloaded*) e será executada numa posterior abertura do VB com o aviso "*xxxxx add-in could not be loaded, do you want to remove it from the list of add-ins*?"
Após isto, você terá que executar novamente o add-in para registrá-lo como relacionado ao VB. Vejamos dois casos destes erros:
1) Referenciando uma propriedade da VBIDE Instance Object, como *AcitveProject.FileName* no evento *ConnectAddin* da *Conector Class*.
2) Conectando mais menus ou submenus que você desconectou.

Programação é, predominantemente, uma ciência exata e muitas "regras não documentadas" são uma real necessidade ao se criar um add-in.
*Por Les Smith**

76 - VB4 - Evitando Erros de Atualização em Bases Access

Evite o erro de acesso 3260 ("*Couldn't update; currently record is locked by user '<userName>' on machine '<userMachineID>*'"), que ocorre quando duas ou mais aplicações acessam a mesma tabela de uma base de dados Access (Jet). O acesso é realizado por objetos *recordset* (como *table* ou *dynaset*), sobre uma tabela que contenha uma chave (primária ou não).
Se uma das aplicações está ociosa (apenas abriu o registro e ainda não o alterou) e outra aplicação tenta alterar ou adicionar dados, o erro citado ocorre. Para evitar este problema, inclua o método *Idle dbFreeLOcks* após o recordset ser aberto. Se for um *table recordset*, inclua-o após ter definido a propriedade Index. Veja um exemplo:

```
Set db = Workspaces(0).Opendatabase("Test.mdb")
Set TB = Db.OpenRecordSet("Customer_Master", dbOpenTable)
TB.Index = "PrimaryKey"
DB.Engine.Idle (dbFreeLocks)
```

*Por Rajesh Patil**

77 - VB4 - Descarregando DLLs fora de controle

Quando uso VB em Windows 95, às vezes ocorre que, um programa torne o sistema operacional instável. Costumava derrubar o Windows 95 e reiniciar para limpar a memória de todos os VBXs e DLLs. Mas descobri, recentemente, uma forma mais prática:
Criei um arquivo DOS Batch chamado RESTART.BAT, no seu disco rígido, com este conteúdo:

```
EXIT
```

No Windows 95, criei um atalho para este BAT. O modo DOS é selecionado, nas propriedades, como *Program / Advanced*. Este caminho é muito mais rápido que um *reboot*.
*Por Michael J. Dyer**

78 - VB3/VB4 - Movendo itens em uma list box

Para, através do mouse, mover a localização de um item numa list box, use o código abaixo.

```
'declarations:
Dim Tmp_Text As String
Dim Old_index As Integer
Dim New_index As Integer
'mouse events:
Sub List1_MouseDown (Button As Integer, _
        Shift As Integer, X As Single, Y As Single)
        Old_index = List1.ListIndex
        Tmp_text = List1.text
End Sub
Sub List1_MouseUp (Button As Integer, _
        Shift As Integer, X As Single, Y As Single)
        New_index  = List1.ListIndex
        If Old_index <> New_index Then
                List1.RemoveItem Old_Index
                List1.AddItem TmpText, NewIndex
        End If
End Sub
```

*Por Márcio Cristiano de Castro Scotti**

79 - VB3/VB4 - Sub Main, iniciando um projeto sem interface

Nem sempre é necessário ou útil que um sistema (projeto VB) inicie com uma janela (*form*). Uma alternativa, nem sempre utilizada, é o de iniciar o projeto executando uma rotina sem form. Alguns exemplos:

1) Processo de inicialização muito longo, com abertura de banco de dados, testes para verificar se o sistema pode ou não ser inicializado, leitura de *Registry* ou arquivos INI, etc.
2) Quando o sistema rodará em "background", ou seja, em segundo plano. Este sistema não terá janelas (forms) para interação com o usuário.

No Visual Basic, o recurso é bastante simples. Primeiro, se informa ao VB que o projeto iniciará com uma **Sub Main** (esta informação é passada nas opções de projeto - *Options* → *Project* → *Start Up Form* = *Sub Main*.) Depois, basta incluir em um dos módulos de código (.BAS), uma rotina com este nome.

```
Sub Main
        'rotinas de inicialização e leitura de opções
```

```
        'logon, abertura de banco de dados e restante da inicialização
        'mostra o primeiro form
                form1.show
End Sub
```

Por Charles A. Müller.

80 - VB3/VB4 - Capturando parâmetros

Todos se recordam das velhas linhas de comando nos programas DOS: *dir /s, pkunzip -v, del /p, mysys /?, myeditor myfile.txt* entre outros. Com estes recursos, os programas já sabiam o que fazer ao serem chamados. No Windows, este recurso ainda é utilizado, principalmente quando se deseja automatizar tarefas. O próprio *Registry* do Windows passa os devidos parâmetros ao executar uma aplicação associada a um tipo de arquivo. Assim, aplicativos da linha Office podem ser disparados para abertura e impressão de arquivos ou execução de macros. Para "capturar" a linha de comando com os parâmetros de um executável em VB, basta utilizar a função *Command*:

```
Select Case UCase(Trim(Command$))
        Case "/A"
                frmAvanc.Show 'usuários avançados
        Case "/M"
                frmMedios.Show 'usuários médios
        Case Else
                frmBasico.Show 'default, usuários novatos
End Select
```

A captura de parâmetros ocorre normalmente na *sub main* ou num evento *load* do form inicial.

Por Charles A. Müller.

81 - VB3/VB4 - Onde está o fim?

Uma aplicação do Visual Basic pode ser encerrada de várias formas: 1) com o fechamento (*unload*) do form principal. 2) com o comando *Stop* (apenas como interrupção na depuração). 3) com o comando *End*. Este último, fecha todos os arquivos e limpa todas as variáveis. O problema do *End* é que esta palavra faz parte de outros comandos como *End Sub* e *End If* (fechamento de blocos). Imagine se você precisar depurar um programa para descobrir quais os pontos em que ele é encerrado, como diferenciar o *End* "puro" dos outros? Basta chamar sempre uma função pública (codificada em um .BAS) que "substituirá" o *End*. Somente esta função terá *End*, facilitando o controle do código.

```
Public Sub SU_AbortaSis
        End 'único local para o End
End Sub
Private Sub Form_Unload (Cancel As Integer)
        ...
        SU_GravaConfiguracoes
```

```
        SU_AbortaSis
End Sub
```
Por Charles A. Müller.

82 - VB3/VB4 - F1 e o Help de Contexto

As aplicações Windows acessam Help (ajuda) diretamente através do pressionamento da tecla F1. Muitas porém, utilizam menus (?→ Conteúdo) ou botões (Ajuda). Nestes casos deve aparecer a página de ajuda indicada na propriedade *HelpContextID* do *form* ou controle. A solução ao programador pode ser o uso de API, com a função WinHelp, na biblioteca User (como sugere o exemplo SAMPLES\ICONWRKS). Uma forma mais simples, é simular o pressionamento de F1:

```
Sub AjudaConteudo_Click ()
        SendKeys "{F1}"
End Sub
```
Por Charles A. Müller.

83 - VB3/VB4 - Validando CGC e CPF

Essa é **brasileiríssima**. Os números de CGC e CPF possuem dígitos verificadores para... adivinhem ... verificar!
Isto é obvio. A validação deve ser feita (por qualquer sistema decente) para impedir a digitação por engano e os CGCs e CPFs falsos ("que coisa feia, tentando passar a perna na gente"). A função abaixo não é de minha autoria, mas, achei no meu "baú" de código.

```
Function Fu_consistir_CgcCpf (Vl_CgcCpf As String)
   ' Esta Rotina Devolverá True  Se o Cgc/Cpf Informado For valido
   '                 ou False Se o Cgc/Cpf Não For Correto
   ' Para Chamar esta Rotina de Consistência
   ' 1 ) Atribuir o valor do CgcCpf a uma Variavel String
   ' 2 ) Chamar a Rotina com : Fu_consistir_CgcCpf (Variavel)
   ' Uma Forma Simples de fazer a Consistencia
         ' é Copiando as linhas abaixo (exemplo)
   ' para dentro do Programa
   ' Dim Vl_CgcCpf As String
   ' Vl_CgcCpf = Me.CgcCpf.Text
   ' If Fu_consistir_CgcCpf(Vl_CgcCpf) = False then
   '  MsgBox "( Cgc/Cpf Informado Não é um Cgc/Cpf Correto )"
   '  Me.CgcCpf.SetFocus
   '  Exit Sub
   ' End if

         Fu_consistir_CgcCpf = False
   Dim VA_CgcCpf          As String
   Dim VA_Digito          As String
```

```
Static Numero(15)       As Integer
Dim VA_Resto            As Integer
Dim VA_Resultado        As Integer
Dim VA_SomaDigito10     As Integer
Dim VA_resto1           As Integer

VA_CgcCpf = Format(Vl_CgcCpf, "@@@@@@@@@@@@@@")
VA_Digito = Mid(VA_CgcCpf, 13, 2)

Numero(1) = Val(Mid(VA_CgcCpf, 1, 1))
Numero(2) = Val(Mid(VA_CgcCpf, 2, 1))
Numero(3) = Val(Mid(VA_CgcCpf, 3, 1))
Numero(4) = Val(Mid(VA_CgcCpf, 4, 1))
Numero(5) = Val(Mid(VA_CgcCpf, 5, 1))
Numero(6) = Val(Mid(VA_CgcCpf, 6, 1))
Numero(7) = Val(Mid(VA_CgcCpf, 7, 1))
Numero(8) = Val(Mid(VA_CgcCpf, 8, 1))
Numero(9) = Val(Mid(VA_CgcCpf, 9, 1))
Numero(10) = Val(Mid(VA_CgcCpf, 10, 1))
Numero(11) = Val(Mid(VA_CgcCpf, 11, 1))
Numero(12) = Val(Mid(VA_CgcCpf, 12, 1))
Numero(13) = Val(Mid(VA_CgcCpf, 13, 1))
Numero(14) = Val(Mid(VA_CgcCpf, 14, 1))

If Len(Trim(Vl_CgcCpf)) > 11 Then  ' Cgc
   VA_Resultado = Numero(1) * 2
   If VA_Resultado > 9 Then
      VA_SomaDigito10 = VA_Resultado + 1
   Else
      VA_SomaDigito10 = VA_Resultado
   End If
   VA_Resultado = Numero(3) * 2
   If VA_Resultado > 9 Then
      VA_SomaDigito10 = VA_SomaDigito10 + VA_Resultado + 1
   Else
      VA_SomaDigito10 = VA_SomaDigito10 + VA_Resultado
   End If
   VA_Resultado = Numero(5) * 2
   If VA_Resultado > 9 Then
      VA_SomaDigito10 = VA_SomaDigito10 + VA_Resultado + 1
   Else
      VA_SomaDigito10 = VA_SomaDigito10 + VA_Resultado
   End If
   VA_Resultado = Numero(7) * 2
   If VA_Resultado > 9 Then
      VA_SomaDigito10 = VA_SomaDigito10 + VA_Resultado + 1
   Else
      VA_SomaDigito10 = VA_SomaDigito10 + VA_Resultado
   End If
   VA_SomaDigito10 = VA_SomaDigito10 + Numero(2) + Numero(4) + Numero(6)
   If Mid(Str(VA_SomaDigito10), Len(Str(VA_SomaDigito10)), 1) = "0" Then
```

```
      VA_Resto = 0
   Else
      VA_Resto = 10 - Val(Mid(Str(VA_SomaDigito10), _
                                          Len(Str(VA_SomaDigito10)), 1))
   End If
   If VA_Resto <> Numero(8) Then
      Exit Function
   End If
   VA_Resultado = (Numero(1) * 5) + (Numero(2) * 4) _
                  + (Numero(3) * 3) + (Numero(4) * 2)  _
                  + (Numero(5) * 9) + (Numero(6) * 8) +  _
                  (Numero(7) * 7) + (Numero(8) * 6) +  _
                  (Numero(9) * 5) + (Numero(10) * 4) + _
                  (Numero(11) * 3) + (Numero(12) * 2)
      ' Atribui para resto o resto da divisão
                  ' de VA_resultado dividido por 11
                  VA_Resto = VA_Resultado Mod 11
                  If VA_Resto < 2 Then
      VA_resto1 = 0
   Else
      VA_resto1 = 11 - VA_Resto
   End If
   If VA_resto1 <> Numero(13) Then
      Exit Function
   End If
   VA_Resultado = (Numero(1) * 6) + _
                  (Numero(2) * 5) + (Numero(3) * 4) + _
                  (Numero(4) * 3) + (Numero(5) * 2) +  _
                  (Numero(6) * 9) + (Numero(7) * 8) + _
                  (Numero(8) * 7) + (Numero(9) * 6) + _
                  (Numero(10) * 5) + (Numero(11) * 4) + _
                  (Numero(12) * 3) + (Numero(13) * 2)
   ' Atribui para resto o resto da divisão
                  ' de VA_resultado dividido por 11
                  VA_Resto = VA_Resultado Mod 11
   If VA_Resto < 2 Then
      VA_resto1 = 0
   Else
      VA_resto1 = 11 - VA_Resto
   End If
   If VA_resto1 <> Numero(14) Then
      Exit Function
   End If
Else  ' Cpf
   VA_Resultado = (Numero(4) * 1) + _
                  (Numero(5) * 2) + (Numero(6) * 3) _
                  + (Numero(7) * 4) + (Numero(8) * 5) _
                  + (Numero(9) * 6) + (Numero(10) * 7)_
                  + (Numero(11) * 8) + (Numero(12) * 9)
   VA_Resto = VA_Resultado Mod 11
   If VA_Resto > 9 Then
```

```
      VA_resto1 = VA_Resto - 10
    Else
      VA_resto1 = VA_Resto
    End If
    If VA_resto1 <> Numero(13) Then
      Exit Function
    End If

    VA_Resultado = (Numero(5) * 1)  _
                   + (Numero(6) * 2) + (Numero(7) * 3) _
                   + (Numero(8) * 4) + (Numero(9) * 5) + _
                   (Numero(10) * 6) + (Numero(11) * 7) + _
                   (Numero(12) * 8) + (VA_Resto * 9)
    VA_Resto = VA_Resultado Mod 11
    If VA_Resto > 9 Then
      VA_resto1 = VA_Resto - 10
    Else
      VA_resto1 = VA_Resto
    End If
    If VA_resto1 <> Numero(14) Then
      Exit Function
    End If

  End If
Fu_consistir_CgcCpf = True
End Function
```

Por Chales A. Müller

84 - VB3/VB4 - Performance com a SQL Passthrough

Quando você acessa uma base dados via ODBC (*Open Database Connectivity*), os drivers ODBC atuarão como **tradutores** dos seus comandos **SQL**. A razão disto é que, existe uma linguagem SQL genérica (SQL ANSI) e dialetos SQL distintos nos vários produtos (linguagens e bancos) disponíveis no mercado. Assim, cada fornecedor de banco de dados poderá incluir recursos (como *storned procedures*) e sintaxes específicas em seus produtos; existem o SQL da Oracle, o SQL da Informix, o SQL da Sybase etc. Escrevendo seus comandos em SQL ANSI, o ODBC irá "interpretar", em tempo de execução, os comandos para a sintaxe SQL do banco que seu usuário acessa. Esta operação tem uma vantagem e uma desvantagem:

1) A vantagem é que um só aplicativo, a priori, poderá ser executado - sem alteração de fontes - em qualquer banco de dados Client Server, pelo padrão ODBC. Além da **portabilidade** de código fonte, existe o ganho em **interoperabilidade**: o programa poderá acessar, ao mesmo tempo, bases diferentes. A interoperabilidade é necessária em empresas, por exemplo, que passaram por processos de fusão ou incorporação com outra empresa (que usa outra "marca" de banco de dados).
2) Desvantagem: a "tradução" impacta consideravelmente na performance do sistema, o aplicativo (que pode estar rodando em uma grande rede) tornar-se-á muito mais lento.

A solução é pedir ao ODBC que "pule" a tradução que seria realizada pelos seus drivers. Assim, ganha-se tempo de execução. Veja este exemplo:

```
Dim VA_Cmd As String 'comando SQL
Dim snapCidade As Snapshop
Dim VA_Cod As Integer 'código da cidade (campo chave)
Const SQLPASSTRHOUGH = 64
'...
VA_Cmd = "Select Cidade, Nome from CIDADE where Cidade = " & VA_Cod
Set snapCidade = db.CreateSnapshop(VA_Cmd, SQLPASSTRHOUGH)
```

A **SQL Passthrough** é o parâmetro para "pular" a tradução. No VB4, a constante chama-se ***dbSQLPassThrough***.

O comando SQL passado deve estar na sintaxe **específica** do SGBD (ou DBMS) utilizado. Mesmo assim, o sistema poderá **continuar como portável e interoperável**, seguindo-se os passos abaixo (código **parametrizado**):

1) Programe todas as consultas em todos os dialetos SQL utilizados pelos seus usuários, escreva o código de um modo fácil de ser compreendido e alterado.
2) Execute a consulta específica do banco *tal* no momento *tal*. A informação de *qual banco* poderá estar em entradas de arquivos INI ou no Registry.

Por Charles A. Müller

85 - VB4 - Listas erradas de API

Os utilitários APILOD16.EXE e APILOD32.EXE acessam o arquivo WIN32API para passar os parâmetros de tipos de dados (Type Declarations) necessários para chamar funções Win32 API. Porém existem erros. Por exemplo:

WIN32API.TXT (incorretamente)diz:

```
Type COMSTAT
        fCtsHold As Long     'errado
        fDsrHold As Long     'errado
        fRlsHold As Long     'errado
        fXoffHold As Long    'errado
        fXoffSnet As Long    'errado
        fEof As Long         'errado
        fTxim As Long        'errado
        fReserved As Long    'errado
        cbInQue As Long
        cbOutQue As Long
End Type
```

WINT31APITXT, corretamente, diz:

```
Type COMSTAT
        bunch_Of_Bits As Long
        cbInQue As Long
        cbOutQue As Long
End Type
```

*Por Andy Rosa**

86 - VB3/VB4 - Centralizando Forms (I)

Para mostrar as janelas no meio da tela, podem ser utilizadas estas rotinas. Quando se deseja centralizar o próprio form, o parâmetro será a palavra *Me* e a rotina será chamada do evento *Form_Load*.

```
Sub CenterForm (f As Form)
   Screen.MousePointer = 11
   f.Top = (Screen.Height) / 2 - f.Height / 2
   f.Left = Screen.Width / 2 - f.Width / 2
   Screen.MousePointer = 0
End Sub
```

Para um **suave deslocamento** do centro, interessante para forms modais, utilize apenas 85% da medida *Height* da tela:

```
f.Top = (Screen.Height * .85) / 2 - f.Height / 2
```

Exemplo de aplicação:

```
Form1.Load
CenterForm Form1
Form1.Show
```

Outro exemplo:

```
Sub Form_Load ( )
        CenterForm Me
        '...
End Sub
```

Para centralizar um form não em relação a tela, mas a outro form, utilize a rotina abaixo. Ela é útil quando há um form principal do sistema, que geralmente é um *MDIForm*.

```
Sub SU_CenterChild (f As Form)
        'centraliza um form dentro do MDIform  (chamado aqui de F00)
        Dim VA_X, VA_Y
  VA_X = (((F00.ScaleWidth - f.Width) \ 2) + F00.Left)
  VA_Y = (((F00.ScaleHeight - f.Height) \ 2) + F00.Top)
  f.Move VA_X, VA_Y
End Sub
```

Por Charles A. Müller

87 - VB4 - Centralizando Forms (II - A versão)

A *dica anterior* mostra como centralizar forms no VB3. A dica também é aplicável ao VB4. Abaixo, há uma outra versão desta rotina. Ela usará um parâmetro opcional (frmParent). O último form lido será centralizado em relação ao "parent" (pai, o principal). Na falta do frmParent, a centralização ocorrerá em relação a tela. Lembramos que esta implementação é somente para a **versão 4** do VB.

```
Public Sub CenterForm(Optional frmParent)
        If Forms.Count = 0 then Exit Sub
        If IsMissing (frmParent) Or Not TypeOf frmParente Is Form  then
                Forms(Forms.Count -1).Move _
                (Screen.Width - Forms(Forms.Count -1).Width / 2, _
```

```
            (Screen.Height - Forms(Forms.Count -1).Height / 2
        Else
            Forms(Forms.Count -1).Move _
            (frmParent.Width - Forms(Forms.Count -1).Width / 2, _
            (frmParent..Height - Forms(Forms.Count -1).Height / 2
        End If
End Sub
```

*Por Denis Basaric**

88 - VB3 - Menu Colar Alternativo

Se você usa alguns controles, como o QuickPack Pro (da Crescent), é impossível atribuir CTRL+V para Editar-Colar. Pois, o texto do Clipboard será colado duas vezes. Para manter a tecla de atalho, atribua **mnuPaste.caption = "Co&lar" + Chr$(9) + "Ctrl + V"**, na Sub Main ou no form_Load.
*Por Daniele Alberti**

89 - VB3/VB4 - Já estou no ar?

Algumas aplicações para Windows podem ter várias instâncias, ou seja, podem ser executadas repetidas vezes ao mesmo tempo no mesmo computador. É o caso do Bloco de Notas, do Paint, da Calculadora e de outros. Existem programas cuja múltipla execução não é interessante, por questões de produtividade ao usuário (como o Word, o File Manager e o Excel) ou segurança (como aplicações que usam banco de dados). Os sistemas comerciais (de banco de dados), em geral, só podem ser executados em uma sessão ao mesmo tempo. O controle disto no VB é feito através do objeto *App*.

```
 Dim SaveTitle as string
 If App.PrevInstance Then
   SaveTitle = App.Title
   App.Title = "... segunda chamada ao mesmo programa."
   Me.Caption = "... segunda chamada ao mesmo programa, serei fechado"
   'se for a Sub Main, a linha acima, obviamente, não existe
   'as linhas abaixo fecham a segunda chamada e alternam para
   'a primeira
   AppActivate SaveTitle
   SendKeys "% R", True
   End
 End If
```

O código acima deve ser a primeira coisa a ser executada na sua aplicação. Assim, ao invés de abrir uma segunda sessão do programa, o Windows irá alternar para a sessão já aberta. Isto também pode ser feito por APIs (*FindWindow*, *ShowWindow* e *SetFocus,* da bilblioteca *User*), mas, tem o mesmo efeito e é mais trabalhoso.
Por Charles A. Müller.

90 - VB3/VB4 - Seja Feliz

Você que já passa horas e horas diante do computador (do VB, do Windows e outros bichos), tire um tempo para um filme, um livro, a família e os amigos.

Até a próxima!
Algumas dicas de Visual Basic foram implementadas em uma versão (3 ou 4) e podem ser utilizadas na outra versão. É necessário, porém, que o leitor observe as pequenas diferenças de sintaxe existentes entre duas versões. Por exemplo, o VB3 não aceita o caracter _ para mudança de linha.
As dicas de autor assinalado (*) são uma adaptação do *Visual Basic Programmer's Journal Technical Tips Supplement (3rd Ed., 08/96)*, da *Fawcett Technical Publications* (001-415-833-7100). Acrescentamos tradução e algumas melhorias, além de dicas nossas. Até a próxima!

Gostaria de participar de um próximo guia de dicas? Então envie a sua contribuição para editor@forumaccess.com. As dicas poderão ser de VB3, VB4, VB5 (inclusive CCE), VBScript, VBA, VBA5 e Access.

***Charles A. Müller** (muller@rla14.pucpr.br), de Curitiba, é Editor Adjunto de Visual Basic da Revista Fórum Access, técnico em Processamento de Dados e Acadêmico de Comunicação (PUC PR). Atua como consultor em Internet, Multimídia e Visual Basic.