

1 MANIPULAÇÃO DO BANCO DE DADOS



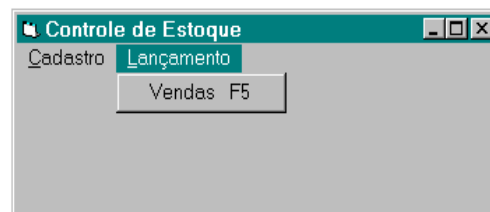
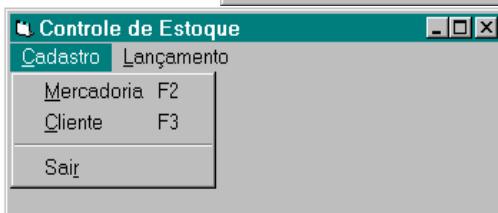
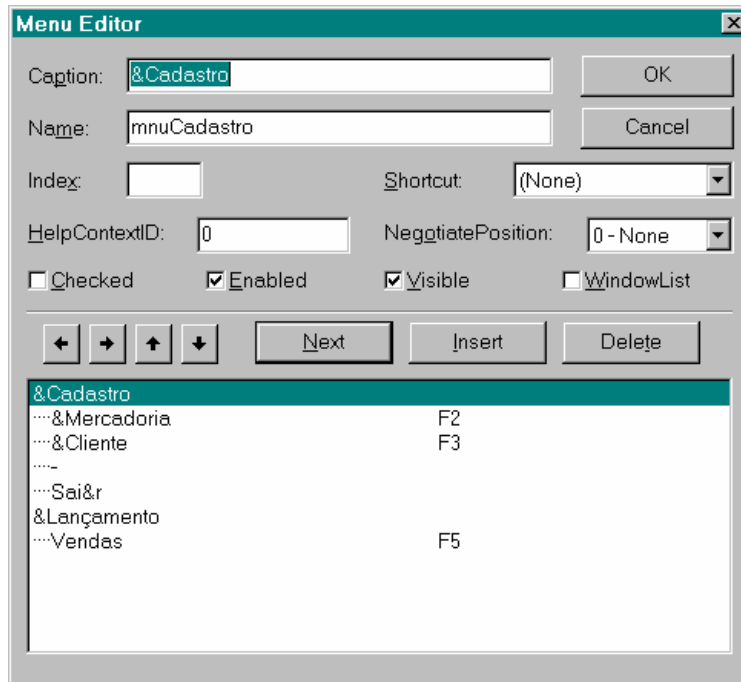
- Abertura do Banco de Dados e suas tabelas
- Adicionar
- Navegação
- Alteração
- Consulta
- Exclusão

1.1 CRIANDO JANELAS

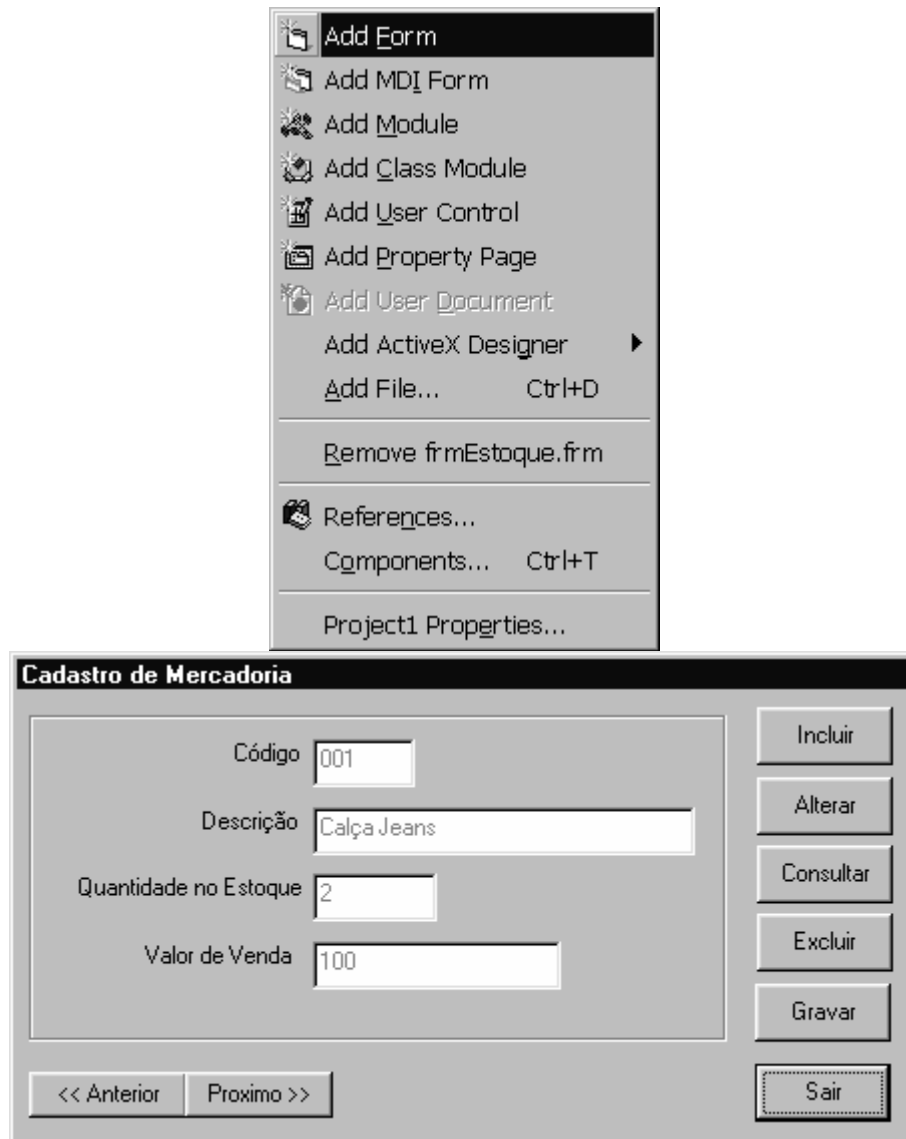
Precisamos criar inicialmente um formulário para cadastrar mercadoria e depois outro formulário para cadastrar o cliente. Vamos usar o que já aprendemos anteriormente:

Crie um novo projeto e para o formulário nomeie-o “frmEstoque”. O Caption será “Controle de Estoque”.

Neste formulário vamos criar os menus que irão acessar nossas bases de dados. Veja como ficará:



No menu "Project" do Visual Basic click na opção "AddForm" para inserirmos um novo formulário em nosso projeto. Será o formulário para cadastramento de mercadoria.



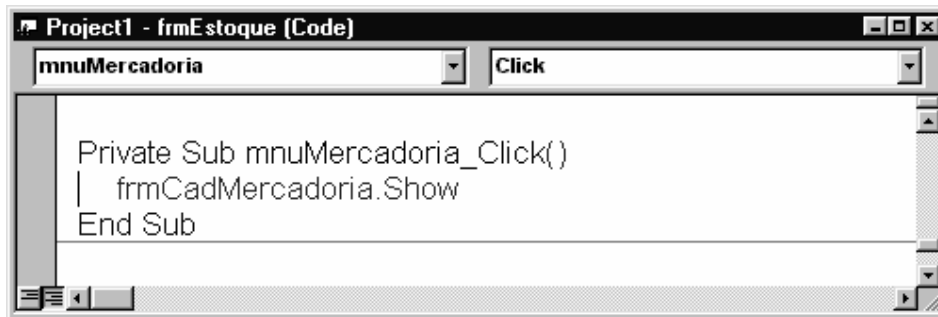
Neste novo formulário altere o **Caption** para “Cadastro de Mercadoria” e o **Name** para “frmCadMercadoria”. Vamos montar esta interface, depois abriremos outros **Form** para Cliente e Vendas.

Coloque um frame, 4 labels e 4 caixa de texto com os **Names**: txtCódigo, txtDescrição, txtQuantidade e txtValor.

Coloque também 7 botões de comando no formulário como na. Para nomea-los use os Captions de cada um prefixado com “cmd”:

Importante: Coloque a propriedade **TabIndex** do Botão "cmdIncluir " como "0" para que ele seja o primeiro a possuir o foco quando entrar na janela.

Para rodar este programa e ver o resultado será necessário ligar este formulário ao menu mnuMercadoria. Para fazer isto vamos até o menu localizado no frmEstoque, click no Cadastro para abrir as opções. Dê dois clickes na opção Mercadoria que uma janela de codificação será aberta. Digite:



```
Project1 - frmEstoque (Code)
mnuMercadoria Click
Private Sub mnuMercadoria_Click()
    frmCadMercadoria.Show
End Sub
```

O Método **Show** abre um novo formulário a partir de outro, e usamos ele para abrir o formulário frmCadMercadoria.

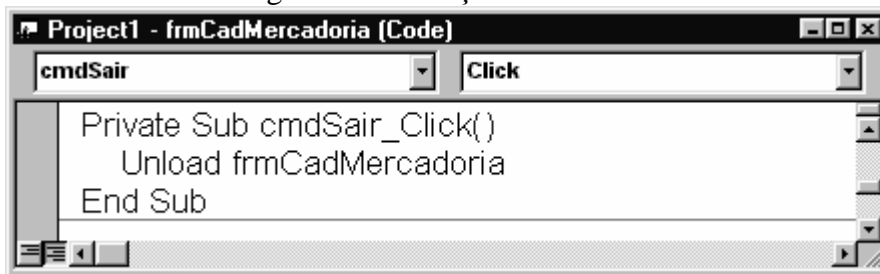
Vamos aproveitar e codificar a opção Sair do menu também. Dê dois clicks nele e digite:



```
Project1 - frmEstoque (Code)
mnuMercadoria Click
Private Sub mnuSair_Click()
    End
End Sub
```

Este simples comando “END” encerra toda a aplicação fechando todas as janelas que podem estar abertas.

Para completar este raciocínio, vamos até o formulário frmCadMercadoria e para o botão cmdSair vamos fazer a seguinte codificação:



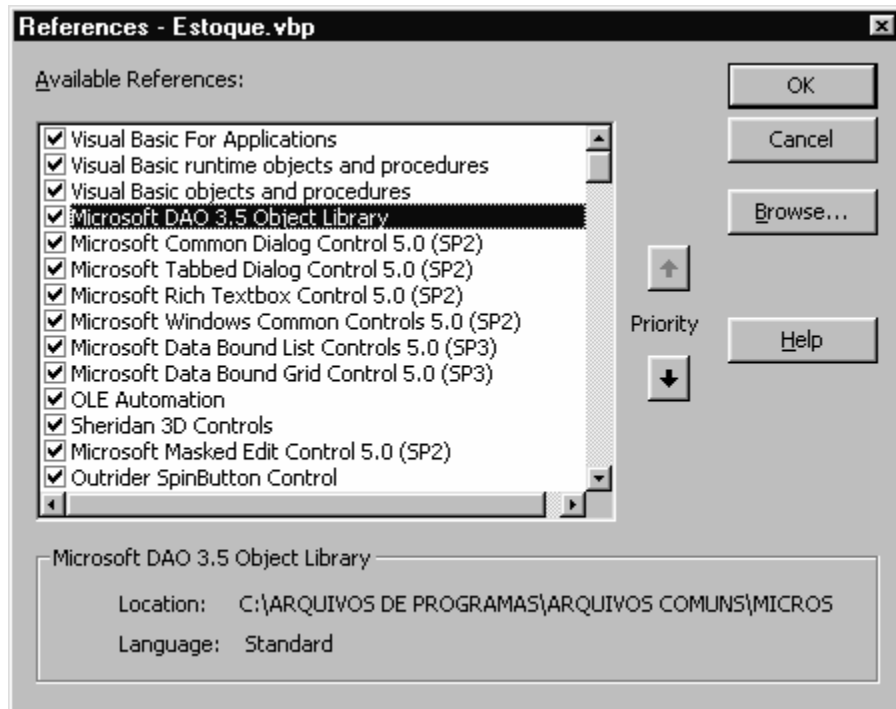
```
Project1 - frmCadMercadoria (Code)
cmdSair Click
Private Sub cmdSair_Click()
    Unload frmCadMercadoria
End Sub
```

O Comando **Unload** retira a janela informada da memória e da tela, voltando a execução do programa para a janela anterior, que no nosso caso é o menu principal.

Agora estamos prontos para começar a codificar o nosso Banco de Dados.

1.1.1 Abrindo um banco de dados

Uma vez que vamos trabalhar com manipulação de dados, temos que verificar se o Visual Basic está apto para fazer esta manipulação. Os serviços necessários para executar essas funções estão em uma DLL do Microsoft DAO 3.5 Object Library. Para que nosso programa leia essa DLL temos que ir até o menu **Project/References** e selecionar esta DLL:

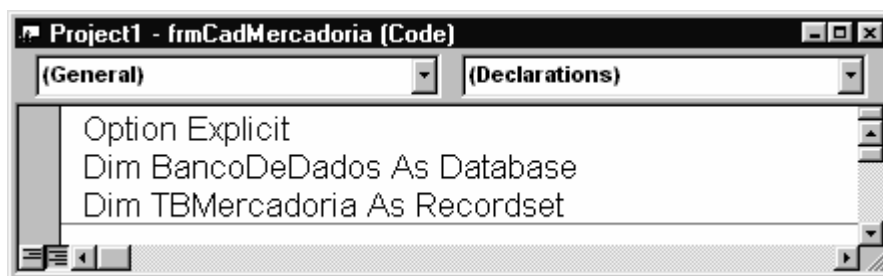


Se este ajuste não for feito e um objeto de banco de dados for declarado, o Visual Basic gerará um erro. A biblioteca DAO (Data Access Objects) fornece um conjunto de objetos de programação que precisaremos usar para gerência os Bancos de Dados.

Temos que criar agora duas variáveis do tipo Banco de Dados e outra do tipo Tabela. Uma servirá para fazer referência ao nome e caminho do Banco de Dados como um todo, e a outra irá fazer referência a uma tabela específica.

Onde criar estas variáveis?

No nosso exemplo temos que criá-las na seção **General** da janela de codificação do formulário. Variáveis criadas ali possuem abrangência em todo o formulário. Se criarmos ela dentro de alguma outra rotina (ou vento) a abrangência destas variáveis serão somente dentro da própria rotina.



A variável BancoDeDados é do tipo Database pois ela sempre terá em seu conteúdo o nome do banco de dados que será aberto. Depois criamos uma variável para a tabela de Mercadoria de nome TBMercadoria.

O Objeto Recordset é a representação lógica de uma tabela. Usaremos este objeto para manipular todos os dados da tabela. Veja as principais propriedades de um Recordset (somente manipulados na codificação do programa):

Sintaxe: *NomeDoRecordSet.Propriedade*

AbsolutePosition : Indica o numero do registro corrente da tabela em uso.

BOF : Retorna True quando a tabela chegou ao inicio e False quando esta em outro ponto.

DateCreated : Retorna a data de criação da tabela manipulada pelo Recordset.

EOF : Retorna True quando a tabela chegou ao fim e False quando esta em outro ponto.

Index : Especificamos o nome do índice que será associado a tabela.

NoMatch : Retorna True se uma pesquisa efetuada dentro da tabela foi bem-sucedida.

PercentPosition : Retorna um percentual referente a posição que a tabela se encontra com comparação com o total de registros da tabela.

RecordCount : Retorna a quantidade de registro que uma tabela possui.

Os principais métodos usamos pelo Recordset:

AddNew : Prepara a tabela para receber um novo registro.

Close : Fecha a tabela, e conseqüentemente o objeto recordset.

Delete : Remove a registro atual da tabela.

Edit : Prepara a tabela para que o registro corrente seja alterado.

FindFirst <Condição> : Procura o primeiro registro dentro da tabela que satisfaça a condição estabelecida (Somente para objeto Recordset tipo dynaset ou snapshot).

FindLast <Condição> : Procura o ultimo registro dentro da tabela que satisfaça a condição estabelecida (Somente para objeto Recordset tipo dynaset ou snapshot).

FindNext <Condição > : Procura o próximo registro dentro da tabela que satisfaça a condição estabelecida (Somente para objeto Recordset tipo dynaset ou snapshot).

FindPrevious <Condição> : Procura o registro anterior dentro da tabela que satisfaça a condição estabelecida (Somente para objeto Recordset tipo dynaset ou snapshot).

MoveFirst : Move a tabela para o primeiro registro.

MoveLast : Move a tabela para o ultimo registro.

MoveNext : Move a tabela para o seguinte seguinte.

MovePrevious : Move a tabela para o registro anterior.

Seek <Operador de Comparação>,<Item a Procurar> : Localiza um registro dentro da tabela. A tabela tem que estar indexada. (Somente para objeto Recordset tipo table)

Update : Grava as alterações e inclusões na tabela.

O próximo passo será abrir o banco de dados e abrir a tabela de mercadoria. Para fazer isto selecione o formulário frmCadMercadoria e abra a janela de codificação dele criando um evento chamado FORM_LOAD. Este evento é chamado sempre que este formulário é chamado pelo Windows.



```
Project1 - frmCadMercadoria (Code)
Form Load
Private Sub Form_Load()
    Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")
    Set TBMercadoria = BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)
End Sub
```

Associamos a variável **BancoDeDados** ao método **OpenDatabase**, que vai realmente abrir o arquivo na prática. Uma vez aberto ao arquivo, sempre que fizermos referência a ele usaremos a variável **BancoDeDados**.

Na abertura do arquivo usamos o Objeto **App.Path** que retorna o nome do diretório corrente, onde esta localizado o arquivo pretendido. Depois concatenamos este diretório com o nome do arquivo a ser aberto.

Para abrir a tabela usamos o método **OpenRecordSet**, e como argumento informamos o nome da tabela a ser aberta e a constante **dbOpenTable** que informa que o RecordSet a ser aberto é do tipo tabela.

Note que é necessário antes do método **OpenRecordSet** usar o nome do objeto Database **BancoDeDados**, pois, dentro da hierarquia, a tabela esta dentro do Banco de Dados.

Fizemos tudo isto para deixar disponível para uso o banco de dados **Estoque** e sua tabela **Mercadoria**.

1.1.2 Abrindo um indice

Próximo passo é abrir o índice que esta tabela irá trabalhar. Este índice é necessário para ordenar os dados que forem digitados num determinada ordem (no nosso caso a ordem é por código) e fazer uma procura rápida pela chave de indexação.

O índice da tabela **Mercadoria** é o **IndCódigo** e possui como chave de indexação o campo **Código**.



```
Project1 - frmCadMercadoria (Code)
Form Load
Private Sub Form_Load()
    Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")
    Set TBMercadoria = BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)
    TBMercadoria.Index = "IndCódigo"
End Sub
```

Usamos a propriedade **Index** para **TBMercadoria** e determinamos que o índice corrente será **IndCódigo**.

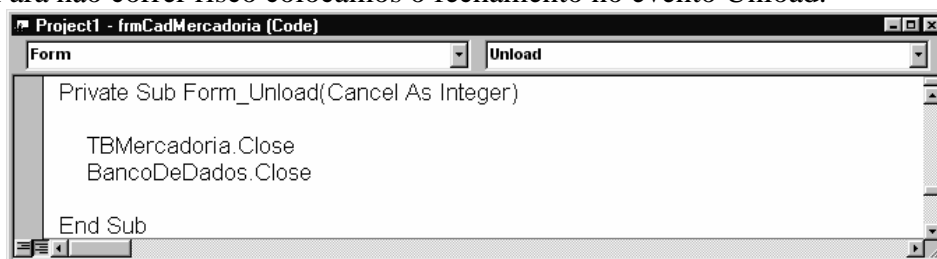
1.1.3 Fechando um banco de dados

É extremamente importante Fechar todas tabelas e banco de dados toda vez que sairmos de um programa ou formulário. Arquivos abertos é perigoso e pode trazer resultados indesejáveis.

Para fecharmos com segurança um banco de dados usamos o Evento do objeto Formulário chamado Unload. Este evento é chamado sempre que o formulário é retirado da tela (e da memória). Note que este evento é o contrário do evento Load, que é lido quando o formulário é colocado na tela.

Porque não podemos fechar os arquivos de Banco de Dados no evento click do botão Sair?

Por que o usuário pode querer inventar de sair deste formulário usando outros recursos do Windows que não seja o botão sair, como a opção Fechar do Control Box, Alt-F4, etc. Para não correr risco colocamos o fechamento no evento Unload.



```
Project1 - frmCadMercadoria (Code)
Form Unload
Private Sub Form_Unload(Cancel As Integer)
    TBMercadoria.Close
    BancoDeDados.Close
End Sub
```

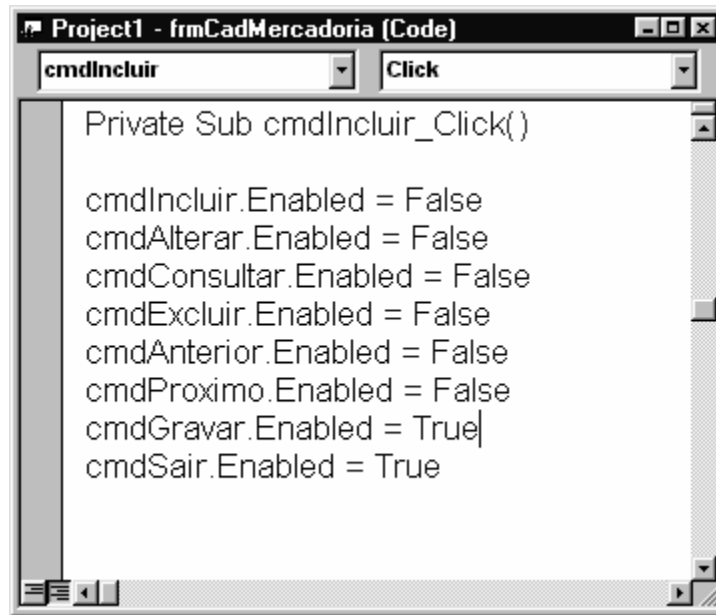
Fechamos primeiro a tabela Mercadoria representada pela variável TBMercadoria e depois o banco de dados Estoque representada pela variável BancoDeDados. Ambas usaram o método *Close*.

1.1.4 Cuidados especiais

Algo que é extremamente importante no desenvolvimento de qualquer programa é tentar prever o máximo possível de erros que o usuário pode cometer e tentar previni-los.

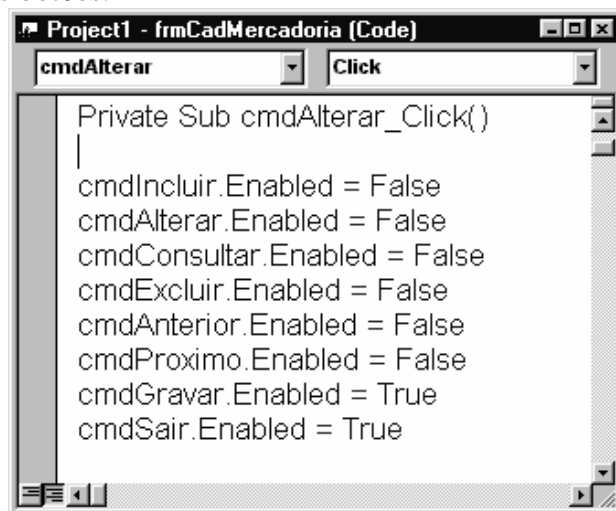
Imagine a situação: o usuário esta no meio de uma inclusão e resolve apertar a tecla excluir ou alterar, sendo que nem terminou a inclusão. Ou então esta numa alteração e resolve apertar o botão alterar novamente. Para podermos tentar cercar esses erros usamos a propriedade **Enabled** nos botões.

Quando o usuário estiver numa inclusão somente os botões Gravar e Cancelar deve estar habilitado. Ou seja, ou o usuário confirma o que esta fazendo ou cancela. Se esta numa alteração a mesma coisa ocorre. E assim por diante. Para fazermos isto temos que usar o **Enabled** em cada botão como veremos abaixo:



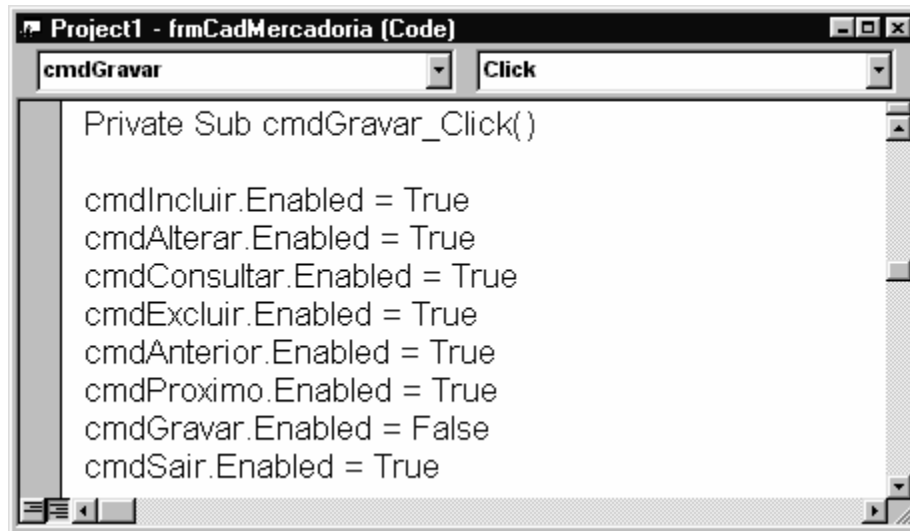
```
Project1 - frmCadMercadoria (Code)
cmdIncluir Click
Private Sub cmdIncluir_Click()
    cmdIncluir.Enabled = False
    cmdAlterar.Enabled = False
    cmdConsultar.Enabled = False
    cmdExcluir.Enabled = False
    cmdAnterior.Enabled = False
    cmdProximo.Enabled = False
    cmdGravar.Enabled = True
    cmdSair.Enabled = True
End Sub
```

Fazendo isto quando o usuário apertar o botão incluir, os botões que ele não pode pressionar enquanto não concluir a inclusão ficarão desabilitados. Deve-se fazer isto para todos os botões.



```
Project1 - frmCadMercadoria (Code)
cmdAlterar Click
Private Sub cmdAlterar_Click()
    cmdIncluir.Enabled = False
    cmdAlterar.Enabled = False
    cmdConsultar.Enabled = False
    cmdExcluir.Enabled = False
    cmdAnterior.Enabled = False
    cmdProximo.Enabled = False
    cmdGravar.Enabled = True
    cmdSair.Enabled = True
End Sub
```

Na alteração o método é semelhante a inclusão. Durante a alteração o usuário só terá liberado para ele os botões "Gravar" e "Sair".



```
Project1 - frmCadMercadoria (Code)
cmdGravar Click
Private Sub cmdGravar_Click()
    cmdIncluir.Enabled = True
    cmdAlterar.Enabled = True
    cmdConsultar.Enabled = True
    cmdExcluir.Enabled = True
    cmdAnterior.Enabled = True
    cmdProximo.Enabled = True
    cmdGravar.Enabled = False
    cmdSair.Enabled = True
End Sub
```

No evento click do botão gravar habilite todos novamente. Com exceção do próprio botão gravar que não pode ser habilitado até que se inclua ou altere algo.

Outro detalhe que é bom lembrar é desabilitar o botão Gravar no evento **Form_Load** para que esteja devidamente desabilitado quando entrar na janela de cadastro de mercadoria. Desabilitamos também o **Frame**, pois assim todos os objetos contido dentro dele serão também desabilitados. Fazemos isto para que o usuário não fique "passeando" pelas caixas de texto sem definir antes (através dos botões) o que ele quer fazer.

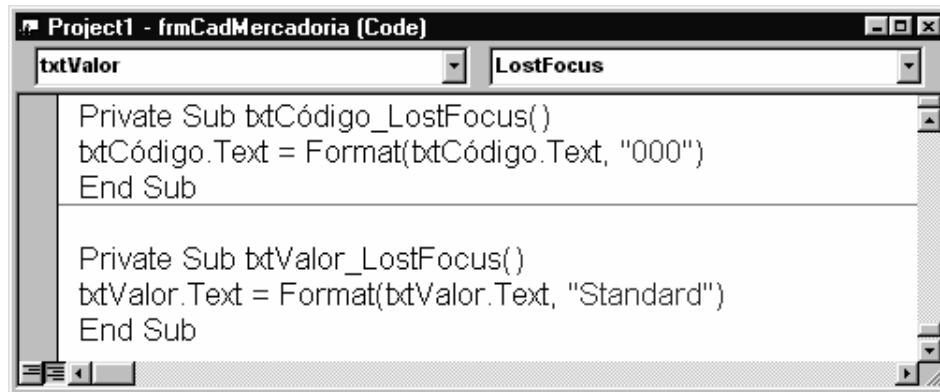


```
Project1 - frmCadMercadoria (Code)
Form KeyPress
Private Sub Form_Load()
    Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")
    Set TBMercadoria = BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)

    TBMercadoria.Index = "IndCódigo"

    cmdGravar.Enabled = False
    Frame1.Enabled = False
End Sub
```

Uma formatação para as caixas de texto que recebem números é bem vindo para dar um aspecto visual mais interessante.

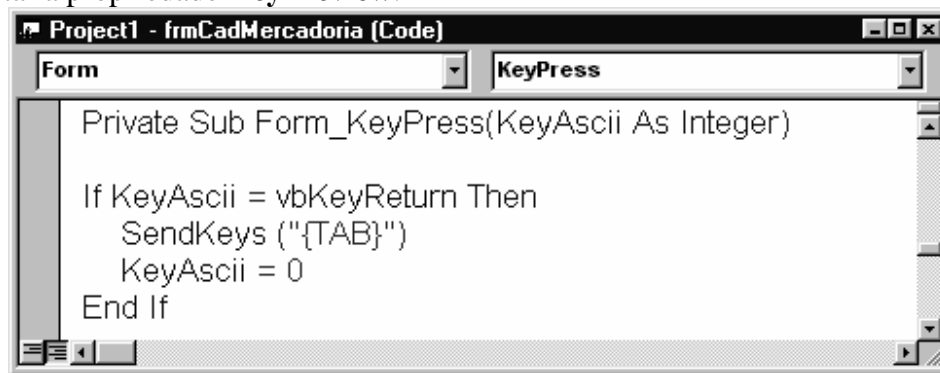


```
Project1 - frmCadMercadoria (Code)
txtValor LostFocus
Private Sub txtCodigo_LostFocus()
    txtCodigo.Text = Format(txtCodigo.Text, "000")
End Sub

Private Sub txtValor_LostFocus()
    txtValor.Text = Format(txtValor.Text, "Standard")
End Sub
```

Criamos uma formatação para as caixas de texto no evento **LostFocus** para quando o usuário terminar de digitar o valor se ajustar.

Habilitamos a tecla Enter criando um evento **KeyPress** no formulário. Não esquecer de habilitar a propriedade **KeyPress**.



```
Project1 - frmCadMercadoria (Code)
Form KeyPress
Private Sub Form_KeyPress(KeyAscii As Integer)

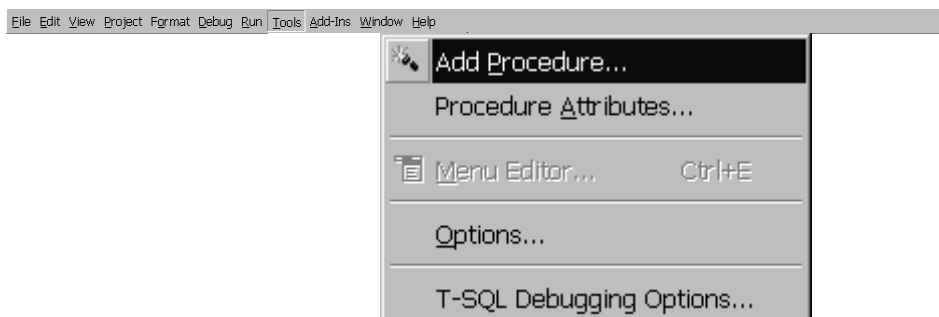
    If KeyAscii = vbKeyReturn Then
        SendKeys (" {TAB} ")
        KeyAscii = 0
    End If

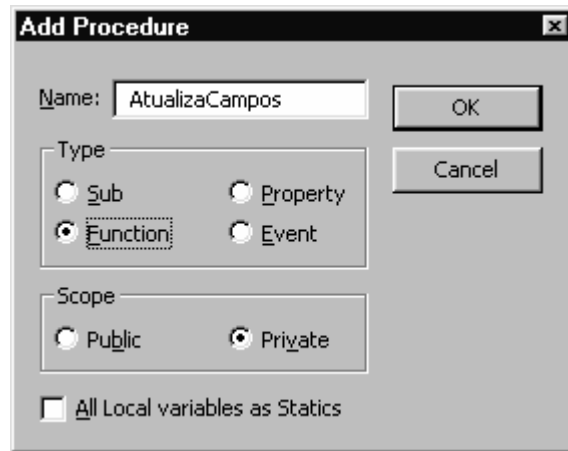
End Sub
```

1.1.5 Funções de apoio

Para que nosso programa de cadastro tenha um código mais limpo e de fácil entendimento vamos criar algumas funções que serão úteis em nossa programação.

Para criar essas funções chame a janela de codificação, selecione **TOOLS** no menu principal do Visual Basic, e escolha a opção **Add Procedure...**





Nomeia a função que estamos criando de AtualizaCampos, coloque do tipo Function e o Scope (abrangência) Private. Com isto, estamos criando uma função que poderá ser usada somente no formulário corrente.

Ao clicar em Ok será aberta uma janela de codificação para podermos digitar o que a função irá conter:

TBMercadoria("Descrição") = txtDescrição
TBMercadoria("Quantidade") = txtQuantidade
TBMercadoria("Valor") = txtValor

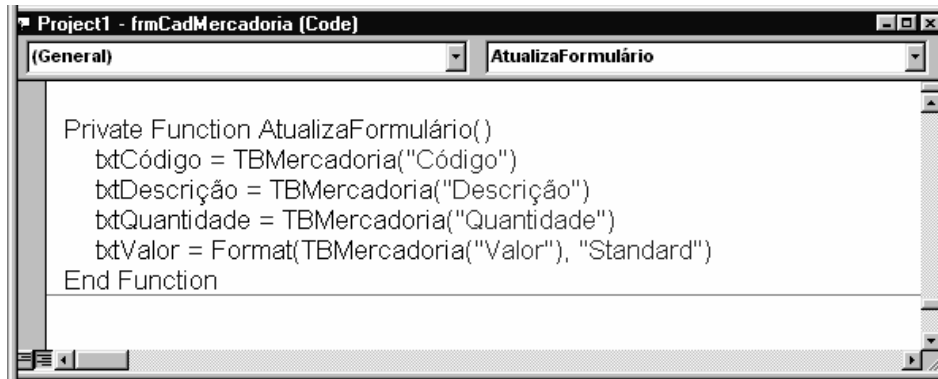
End Function"/>

```
Private Function AtualizaCampos()  
  
TBMercadoria("Código") = txtCódigo  
TBMercadoria("Descrição") = txtDescrição  
TBMercadoria("Quantidade") = txtQuantidade  
TBMercadoria("Valor") = txtValor  
  
End Function
```

A finalidade desta função é inserir o conteúdo existente nas caixas de textos do formulário para dentro dos campos da tabela de mercadoria.

Com isto, sempre que precisarmos gravar os dados que o usuário digitou no formulário para dentro de seus respectivos campos na tabela usamos a função AtualizaCampos.

Repita o processo para criar agora a função AtualizaFormulário:

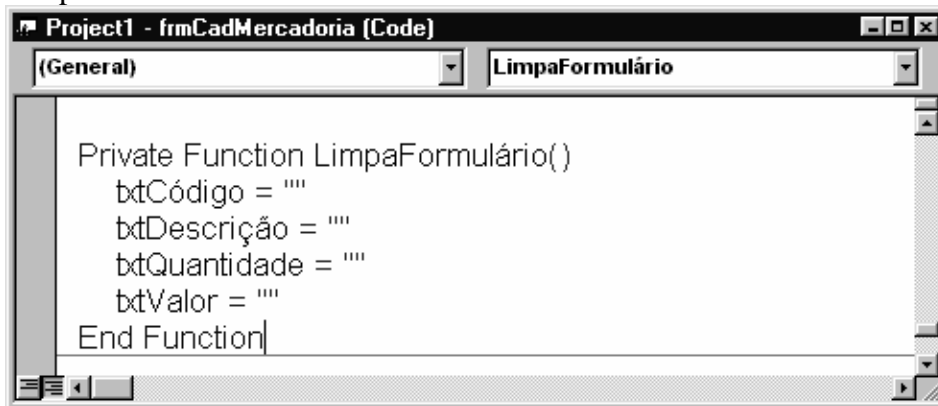


```
Project1 - frmCadMercadoria (Code)
(General) AtualizaFormulário

Private Function AtualizaFormulário()
    btCódigo = TBMercadoria("Código")
    btDescrição = TBMercadoria("Descrição")
    btQuantidade = TBMercadoria("Quantidade")
    btValor = Format(TBMercadoria("Valor"), "Standard")
End Function
```

Note que o `AtualizaFormulário` faz o caminho inverso: Descarrega os dados existente na tabela de seus respectivos campos para dentro das caixas de texto existente no formulário.

Por fim, vamos criar uma função para limpar as caixas de texto. Dê o nome de `LimpaFormulário`:



```
Project1 - frmCadMercadoria (Code)
(General) LimpaFormulário

Private Function LimpaFormulário()
    btCódigo = ""
    btDescrição = ""
    btQuantidade = ""
    btValor = ""
End Function
```

Vamos usar esta função sempre que precisarmos que as caixas de texto fiquem vazias.

1.2 ADICIONANDO DADOS

Antes de começarmos a fazer a codificação para o botão `Incluir`, a fim de preparar nosso formulário para manipular a tabela de `Mercadoria`, vamos fazer uma modificação nas propriedades `MaxLength` das caixas de texto: todas devem possuir o mesmo valor da propriedade `Size` usada para os campos na tabela. Usamos isto para não deixar que um usuário coloque um nome com 40 caracteres num campo que foi definido para receber 30 caracteres.

Outra modificação será efetuada no evento `Form_Load` do formulário `frmCadMercadoria`:

```
Private Sub Form_Load()  
Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")  
Set TBMercadoria = BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)  
  
TBMercadoria.Index = "IndCódigo"  
  
cmdGravar.Enabled = False  
Frame1.Enabled = False  
  
If TBMercadoria.EOF = False Then  
    AtualizaFormulário  
End If  
  
End Sub
```

Quando uma tabela é aberta o Visual Basic se posiciona no primeiro registro existente dentro dela. Entretanto, quando ainda não existe nada digitado dentro da tabela, conseqüentemente não existirá nenhum registro para se posicionar. Neste caso o fim de arquivo é encontrado logo na abertura da tabela.

A propriedade EOF é quem informa se o fim de arquivo esta ativo ou não. Se EOF for True (verdadeiro) significa que a tabela chegou ao fim de arquivo, se for False, significa que a tabela esta posicionada em algum registro em algum lugar.

Recapitulando: A propriedade EOF retorna True se a posição do registro atual for posterior ao último registro da tabela, ou seja, se o fim do arquivo for localizado, e False se a posição do registro atual for no último registro ou anterior a ele.

O Visual Basic possui também a propriedade BOF para informar se o inicio do arquivo foi encontrado.

A propriedade BOF retorna True se a posição do registro atual for anterior ao primeiro registro e False se a posição do registro atual estiver no primeiro registro ou for posterior a ele.

Os valores de retorno das propriedades BOF e EOF são determinados pela localização do indicador do registro atual.

Podemos usar as propriedades BOF e EOF para determinar se um objeto Recordset contém registros ou se você foi além dos limites de um objeto Recordset ao percorrer seus registros.

Perceba que somente se EOF for False que atualizamos o formulário, pois caso contrário, não poderíamos atualizar o formulário pois não haveria dados a serem atualizados.

O primeiro botão que vamos codificar é o incluir. Ele funcionará da seguinte forma: Quando clicarmos nele as caixas de textos serão limpas e o método **AddNew** será acionado. Este método é usado para avisar à tabela que alguns dados serão incluídos.

Depois de usar o **AddNew** temos que atualizar os campos, ou seja, passar tudo que foi digitado nas caixas de texto para seus respectivos campos dentro da tabela TBMercadoria. Fazendo isto usamos o método **Updated** para efetuar a gravação propriamente dita e atualizar a tabela com os novos campos.

Veja um exemplo:

- TBMercadoria.AddNew
- TBMercadoria("Código") = "001"
- TBMercadoria("Descrição") = "Calça Jeans"
- TBMercadoria("Quantidade") = "12"
- TBMercadoria("Valor") = "100"
- TBMercadoria.Updated

Esta rotina é o padrão para inclusão de dados. Repare que sempre inicia com **AddNew** e termina com **Updated** Neste exemplo colocamos expressões string para serem inseridas nos campos, mas poderíamos ter colocado variáveis.

No nosso exemplo vamos colocar as caixas de texto. Ficaria assim:

- TBMercadoria.AddNew
- TBMercadoria("Código") = txtCódigo
- TBMercadoria("Descrição") = txtDescrição
- TBMercadoria("Quantidade") = txtQuantidade
- TBMercadoria("Valor") = txtValor
- TBMercadoria.Updated

Desta forma o que o usuário digitar nas caixas de texto serão incluídos dentro dos campos da tabela.

Isto é uma forma simplificada de fazer inclusão de dados. Mas nós vamos usar um forma mais complexa:

```

Project1 - frmCadMercadoria [Code]
cmdIncluir Click
Private Sub cmdIncluir_Click()

    cmdIncluir.Enabled = False
    cmdAlterar.Enabled = False
    cmdConsultar.Enabled = False
    cmdExcluir.Enabled = False
    cmdAnterior.Enabled = False
    cmdProximo.Enabled = False
    cmdGravar.Enabled = True
    cmdSair.Enabled = True

    LimpaFormulário
    Frame1.Enabled = True
    txtCódigo.SetFocus

End Sub

```

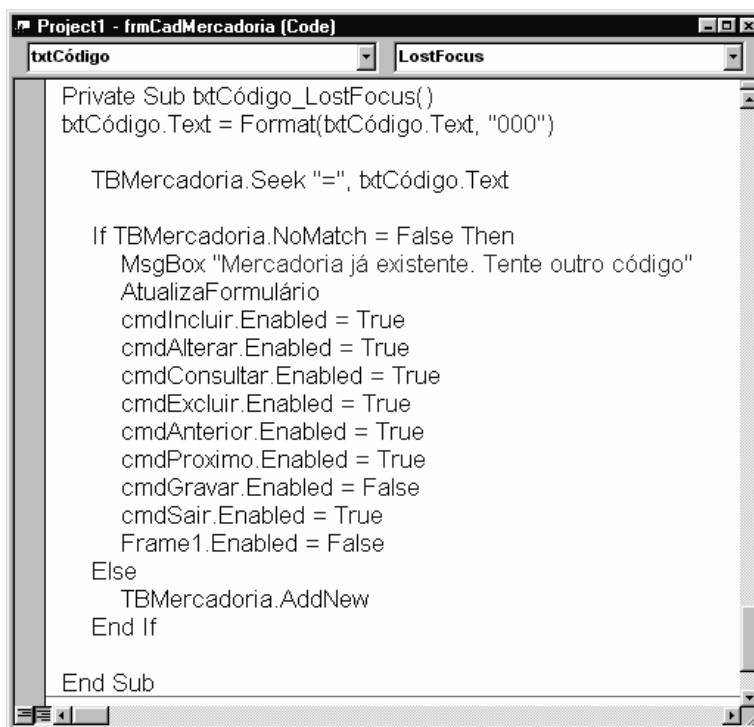
Veja o que foi feito: Sempre que o usuário clicar neste botão o programa limpa o formulário para deixar as caixas de texto vazias para serem preenchidas com novos dados. O **Frame** é habilitado para que as caixas de texto possam ser manipuladas. Em seguida o

foco é passado para o caixa de texto **txtCódigo** pois é a primeira que deve ser digitada pelo usuário

Ainda não aplicamos o método **AddNew** pois ainda não sabemos se o usuário vai incluir um código válido. Primeiro o programa tem que analisar o código que ele digitou, verificar se ele realmente não existe para depois usar o método **AddNew**. Fazemos isto no evento **LostFocus** da Caixa de Texto txtCódigo.

Neste evento verificamos primeiramente se o botão CmdGravar esta habilitado. Se estiver é porque a inclusão foi acionada.

O programa procura o código digitado. Se ele existir sera dado um aviso ao usuario e o processo de inclusão será cancelado, caso contrário (ou seja, se o código não existir dentro da tabela) o método **AddNew** é chamado a fim de preparar a tabela para receber um novo registro.



```
Project1 - frmCadMercadoria (Code)
txtCódigo LostFocus

Private Sub btCódigo_LostFocus()
    btCódigo.Text = Format(btCódigo.Text, "000")

    TBMercadoria.Seek "=", btCódigo.Text

    If TBMercadoria.NoMatch = False Then
        MsgBox "Mercadoria já existente. Tente outro código"
        AtualizaFormulário
        cmdIncluir.Enabled = True
        cmdAlterar.Enabled = True
        cmdConsultar.Enabled = True
        cmdExcluir.Enabled = True
        cmdAnterior.Enabled = True
        cmdProximo.Enabled = True
        cmdGravar.Enabled = False
        cmdSair.Enabled = True
        Frame1.Enabled = False
    Else
        TBMercadoria.AddNew
    End If
End Sub
```

Pronto, a janela já esta preparada para receber os dados. Mas agora falta fazer a codificação para o botão Gravar.

```
Private Sub cmdGravar_Click()

cmdIncluir.Enabled = True
cmdAlterar.Enabled = True
cmdConsultar.Enabled = True
cmdExcluir.Enabled = True
cmdAnterior.Enabled = True
cmdProximo.Enabled = True
cmdGravar.Enabled = False
cmdSair.Enabled = True

Frame1.Enabled = False
txtCodigo.Enabled = True

AtualizaCampos
TBMercadoria.Update

End Sub
```

Atualizamos os campos da tabela com os dados digitado nas caixas de textos e depois chamamos o método **Update** que fará a gravação física dentro do banco de dados.

- Tente incluir vários dados na tabela para podermos usá-los posteriormente nos próximos exercícios.
- Um recurso interessante que pode ser acrescentado é desabilitar os botões de Alteração, Consulta, Exclusão, Anterior e Próximo quando a tabela estiver vazia, pois se não houver nenhum registro, não vai haver nada para alterar, consultar, excluir, etc. Somente o botão de inclusão que poderá ficar habilitado.
- A maneira mais usual de saber se a tabela está vazia ou não é através da propriedade **RecordCount** que informa quantos registros existem gravados dentro da tabela.

1.3 PRÓXIMO E ANTERIOR

Estes botões serão usados o usuário “passar” pela tabela, verificando os dados de todos os registros cadastrados na ordem especificada pelo Index.

The image shows two screenshots of a Visual Basic code editor window titled 'Project1 - frmCadMercadoria (Code)'. The top screenshot shows the code for the 'cmdProximo' button's Click event. The code is as follows:

```
Private Sub cmdProximo_Click()  
    TBMercadoria.MoveNext  
    If TBMercadoria.EOF = True Then  
        TBMercadoria.MovePrevious  
    End If  
    AtualizaFormulário  
End Sub
```

The bottom screenshot shows the code for the 'cmdAnterior' button's Click event. The code is as follows:

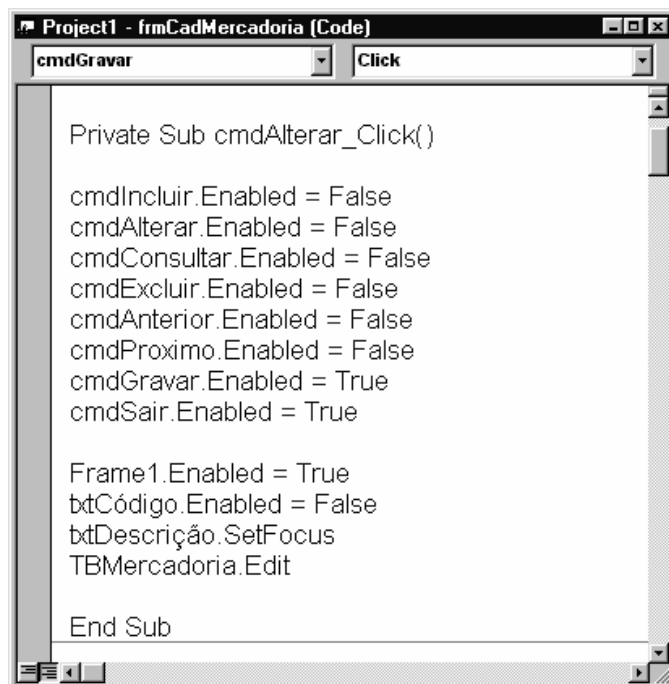
```
Private Sub cmdAnterior_Click()  
    TBMercadoria.MovePrevious  
    If TBMercadoria.BOF = True Then  
        TBMercadoria.MoveNext  
    End If  
    AtualizaFormulário  
End Sub
```

O botão Próximo irá mostrar na tela o próximo registro existente. Note que para fazer este movimento usamos a propriedade MoveNext (mova para o próximo), depois fazemos um teste com a propriedade EOF para verificar se foi movido para o ultimo registro na tabela. Se estiver no último, dali ele não pode passar pois não encontrará nada. Encontrando então o ultimo registro, se tentar passar dali a propriedade MovePrevious (move para o anterior) é acionado. Depois disto atualizamos o formulário.

O botão Anterior possui a mesma lógica, mas invertemos as outras propriedades para fazer agora o teste de inicio de arquivo. Ou seja, se o usuário estiver no primeiro registro ele não poderá voltar um registro, pois antes do primeiro não existira nada.

Se você usar MoveNext quando o último registro for o atual, a propriedade EOF será configurada para True e não haverá registro atual. Se você usar MoveNext novamente, um erro ocorrerá; EOF permanece True. Se recordset referir-se a um Recordset tipo table (que é o nosso caso), a movimentação segue o índice atual. Você pode definir o índice atual usando a propriedade Index. Se você não configurar o índice atual, a ordem de registros retornados será indefinida.

1.4 ALTERAÇÃO



```
Project1 - frmCadMercadoria (Code)
cmdGravar Click
Private Sub cmdAlterar_Click()
    cmdIncluir.Enabled = False
    cmdAlterar.Enabled = False
    cmdConsultar.Enabled = False
    cmdExcluir.Enabled = False
    cmdAnterior.Enabled = False
    cmdProximo.Enabled = False
    cmdGravar.Enabled = True
    cmdSair.Enabled = True

    Frame1.Enabled = True
    txtCódigo.Enabled = False
    txtDescrição.SetFocus
    TBMercadoria.Edit
End Sub
```

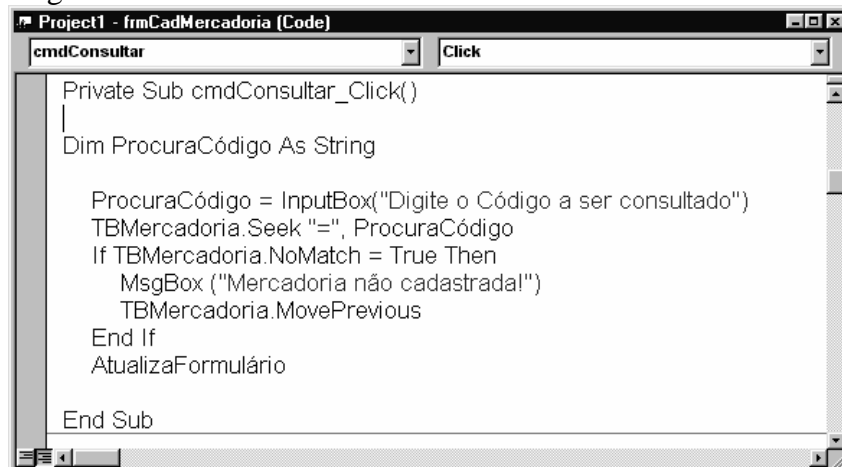
Para alteração vamos fazer algo semelhante ao que fizemos na inclusão. A diferença ficará numa propriedade nova para nós que é EDIT. Esta propriedade abre a tabela para que o dado que esta sendo alterado seja editado.

Note que não liberamos a caixa de texto txtCódigo para alteração, pois sendo ela a chave de índice, não pode ser alterada. Quando necessitar de alteração no código, ele deve ser excluído e incluído novamente.

1.5 CONSULTA

Para consulta vamos usar a função INPUTBOX. Aprendemos a usá-la nos capítulos anteriores. Ela solicita ao usuário a digitação de algum dado e armazena numa determinada variável.

Então criamos uma variável de nome ProcuraCódigo e usamos ela para receber o que o usuário digitar.



```
Project1 - frmCadMercadoria (Code)
cmdConsultar Click
Private Sub cmdConsultar_Click()
    Dim ProcuraCódigo As String

    ProcuraCódigo = InputBox("Digite o Código a ser consultado")
    TBMercadoria.Seek "=", ProcuraCódigo
    If TBMercadoria.NoMatch = True Then
        MsgBox ("Mercadoria não cadastrada!")
        TBMercadoria.MovePrevious
    End If
    AtualizaFormulário
End Sub
```

Feito isto, usamos o método `Seek` para efetuar uma procura dentro da tabela.

Seek: Localiza o registro de um objeto `Recordset` tipo `table` indexado que satisfaça os critérios especificados para o índice atual e torna esse registro o registro atual.

Após o `Seek` colocamos um sinal de comparação para determinar o tipo de procura que este método fara. Podemos usar “=”, “<”, “>”, “<=”, “>=”, etc

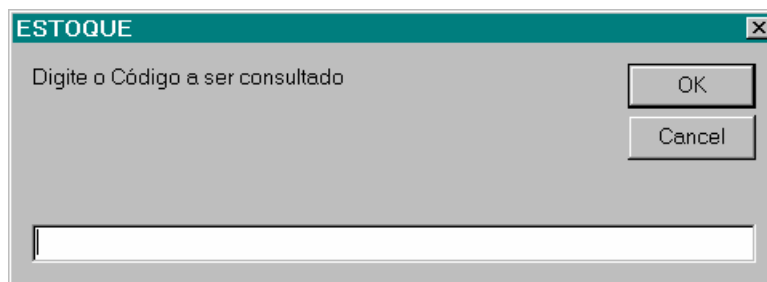
O método `Seek` pesquisa os campos-chave especificados e localiza o primeiro registro que satisfaça os critérios especificados na comparação dentro da chave de índice. Uma vez encontrado, esse registro torna-se o registro atual e a propriedade **NoMatch** é definida como `False`. Se o método **Seek** falhar em localizar um registro correspondente, a propriedade **NoMatch** é configurada como `True` e o registro atual é indefinido.

Se comparação for igual (=), maior ou igual a (>=) ou maior do que (>), o **Seek** começa a pesquisa no início do índice. Se comparação for maior do que (<) ou maior ou igual a (<=), o **Seek** começa a pesquisa no final do índice e continua em direção ao início a menos que existam entradas de índice duplicadas no final. Neste caso, **Seek** começa por uma entrada arbitrária entre as entradas de índice duplicadas no final do índice.

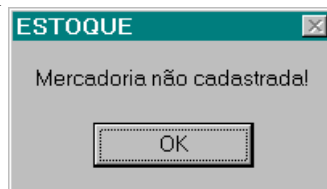
Quando informamos uma chave para o método **Seek** procurar, esta chave deve ser no mesmo formato que o estabelecido no índice do arquivo (`index`). Por exemplo: no `TBMercadoria` a chave do índice é o *Código*, então o *Seek* somente faz a procura pelo código. Não podemos pedir para procurar pela Descrição da mercadoria, pois para isto deveria existir um índice para descrição.

Em nosso exemplo se **NoMatch** for `True` (ou seja, se não encontrou nenhum registro que seja igual ao conteúdo da variável *ProcuraCódigo*) movemos para o registro anterior e depois o formulário é atualizado. Fazemos isto pois quando uma procura não é bem sucedida a tabela é posicionada no fim de arquivo, e como não existe nenhum registro nesta posição, movimentamos para um registro antes, onde o ultimo registro incluído se encontra.

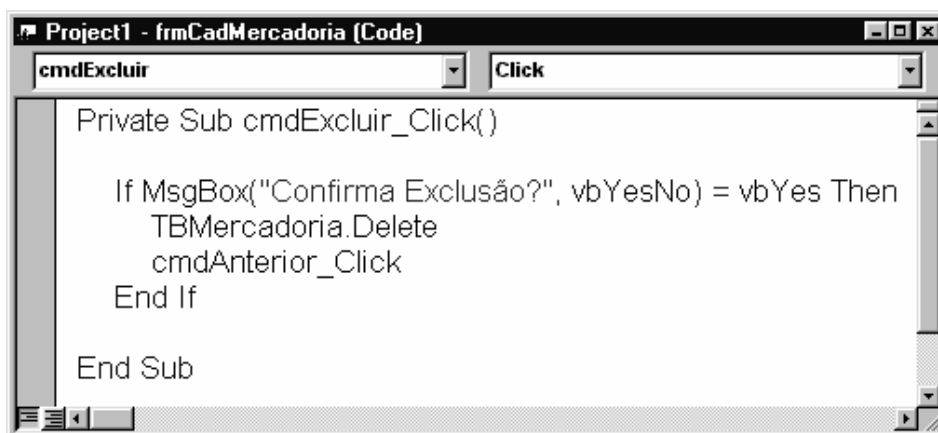
Quando clicarmos no botão Consultar uma janela como a que esta abaixo aparecerá no video:



Digitamos o código que queremos procurar e ao clicar `Ok` a busca é iniciada. Se não encontrar uma mensagem será apresentada:



1.6 EXCLUSÃO



```
Project1 - frmCadMercadoria (Code)
cmdExcluir Click
Private Sub cmdExcluir_Click()
    If MsgBox("Confirma Exclusão?", vbYesNo) = vbYes Then
        TBMercadoria.Delete
        cmdAnterior_Click
    End If
End Sub
```

Quando se vai deletar algo é bom solicitar ao usuário uma confirmação, pois a deleção acidental pode causar resultados catastróficos. Então é bom que o usuário tenha certeza do que está fazendo.

Quando usamos a propriedade *Delete* o registro atual (aquele que está aparecendo no vídeo) é excluído da tabela.

Mas para a tela não ficar vazia (uma vez que aquele registro que estava lá foi eliminado) usamos a procedure *cmdAnterior_click* que é a rotina que usamos quando o botão anterior é acionado.

Fazemos isto para posicionar a tabela no registro anterior ao que foi deletado e mostrá-lo na tela. Desta forma a tela não fica vazia.

Em nosso programa podemos localizar um registro com a consulta ou através dos botões Anterior e Próximo. Localizando podemos clicar no botão Excluir que o registro desaparece.

1.7 CONSIDERAÇÕES FINAIS

Pronto. Está finalizado este formulário *frmCadMercadoria*. Agora usando estes mesmos processos usados aqui programe o Cadastro de Cliente. As vendas é um pouco diferente pois esta tabela depende da tabela de Mercadoria e da tabela de Clientes, ou seja, é necessário, nas vendas, informar quem é o cliente que está comprando e qual mercadoria se está comprando. Estas informações são coletadas em suas respectivas tabelas. Mas vamos aprender a codificá-la após terminarmos com os Clientes.

1.8 CADASTRO DE CLIENTES

Cadastro de Clientes

Código 001

Nome do Cliente Jose da Silva

Limite de Crédito 1400

Incluir

Alterar

Consultar

Excluir

Gravar

Sair

<< Anterior Proximo >>

Option Explicit

Dim BancoDeDados As Database

Dim TBCliente As Recordset

Private Sub cmdAlterar_Click()

cmdIncluir.Enabled = False

cmdAlterar.Enabled = False

cmdConsultar.Enabled = False

cmdExcluir.Enabled = False

cmdAnterior.Enabled = False

cmdProximo.Enabled = False

cmdGravar.Enabled = True

cmdSair.Enabled = True

Frame1.Enabled = True

txtCódCliente.Enabled = False

txtNome.SetFocus

TBCliente.Edit

End Sub

Private Sub cmdAnterior_Click()

TBCliente.MovePrevious

If TBCliente.BOF = True Then

TBCliente.MoveNext

End If

```

        AtualizaFormulário

End Sub

Private Sub cmdConsultar_Click()

Dim ProcuraCódCliente As String

        ProcuraCódCliente = InputBox("Digite o Código do Cliente a ser
consultado")
        TBCliente.Seek "=", ProcuraCódCliente
        If TBCliente.NoMatch = True Then
            MsgBox ("Cliente não cadastrado!")
            TBCliente.MovePrevious
        End If
        AtualizaFormulário

End Sub

Private Sub cmdExcluir_Click()

        If MsgBox("Confirma Exclusão?", vbYesNo) = vbYes Then
            TBCliente.Delete
            cmdAnterior_Click
        End If

End Sub

Private Sub cmdGravar_Click()

cmdIncluir.Enabled = True
cmdAlterar.Enabled = True
cmdConsultar.Enabled = True
cmdExcluir.Enabled = True
cmdAnterior.Enabled = True
cmdProximo.Enabled = True
cmdGravar.Enabled = False
cmdSair.Enabled = True

Frame1.Enabled = False
txtCódCliente.Enabled = True

AtualizaCampos
TBCliente.Update

```



```
End Sub
```

```
Private Sub cmdIncluir_Click()
```

```
cmdIncluir.Enabled = False  
cmdAlterar.Enabled = False  
cmdConsultar.Enabled = False  
cmdExcluir.Enabled = False  
cmdAnterior.Enabled = False  
cmdProximo.Enabled = False  
cmdGravar.Enabled = True  
cmdSair.Enabled = True
```

```
LimpaFormulário
```

```
Frame1.Enabled = True  
txtCódCliente.SetFocus
```

```
End Sub
```

```
Private Sub cmdProximo_Click()
```

```
    TBCliente.MoveNext  
    If TBCliente.EOF = True Then  
        TBCliente.MovePrevious  
    End If  
    AtualizaFormulário
```

```
End Sub
```

```
Private Sub cmdSair_Click()
```

```
    Unload frmCadClientes  
End Sub
```

```
Private Sub Form_KeyPress(KeyAscii As Integer)
```

```
    If KeyAscii = vbKeyReturn Then  
        SendKeys (" {TAB} ")  
        KeyAscii = 0  
    End If
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")
Set TBCliente = BancoDeDados.OpenRecordset("Cliente", dbOpenTable)
```

```
TBCliente.Index = "IndCódigo"
```

```
cmdGravar.Enabled = False
Frame1.Enabled = False
```

```
If TBCliente.EOF = False Then
    AtualizaFormulário
End If
```

```
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
```

```
    TBCliente.Close
    BancoDeDados.Close
```

```
End Sub
```

```
Private Sub txtCódCliente_LostFocus()
    txtCódCliente.Text = Format(txtCódCliente.Text, "000")
```

```
    TBCliente.Seek "=", txtCódCliente.Text
```

```
    If TBCliente.NoMatch = False Then
        MsgBox "Cliente já existente. Tente outro Código"
        AtualizaFormulário
        cmdIncluir.Enabled = True
        cmdAlterar.Enabled = True
        cmdConsultar.Enabled = True
        cmdExcluir.Enabled = True
        cmdAnterior.Enabled = True
        cmdProximo.Enabled = True
        cmdGravar.Enabled = False
        cmdSair.Enabled = True
        Frame1.Enabled = False
```

```
    Else
        TBCliente.AddNew
    End If
```

```
End Sub
```

```

Private Sub txtValor_LostFocus()
txtLimiteCrédito.Text = Format(txtLimiteCrédito.Text, "Standard")
End Sub

Private Function AtualizaCampos()

    TBCliente("CódCliente") = txtCódCliente
    TBCliente("Nome") = txtNome
    TBCliente("LimiteCrédito") = txtLimiteCrédito

End Function

Private Function AtualizaFormulário()

    txtCódCliente = TBCliente("CódCliente")
    txtNome = TBCliente("Nome")
    txtLimiteCrédito.Text = Format(TBCliente("LimiteCrédito"),
"Standard")

End Function

Private Function LimpaFormulário()

    txtCódCliente = ""
    txtNome = ""
    txtLimiteCrédito = ""

End Function

```

1.9 LANÇAMENTO DAS VENDAS

A criação do formulário será como no exemplo acima; ou seja, comparando com os anteriores, a diferença é somente o acréscimo de dois labels que ficarão ao lado do código da mercadoria e do cliente. O primeiro label conterà a descrição da mercadoria, e vamos nomeá-lo para *lblDescrição*, e o segundo terá o nome do cliente, e vamos nomeá-lo para *lblNomeCliente*.

Para esta tabela é necessário criar um índice em que a chave de procura seja o Código da Mercadoria e o Código do Cliente. Usamos para isto o Visual Data Manager:

Note que esses dados não existem na tabela de Vendas. O que existe nesta tabela é somente o código da mercadoria. A descrição está em outra tabela (Mercadoria), assim como o nome do cliente também está em outra tabela (Cliente). Para podermos acessar estas tabelas e capturar somente a descrição da mercadoria e nome do cliente, temos somente um ponto de referência, que é o código deles que existe na tabela de Vendas.

A lógica usada neste programa consiste em pegarmos o código existente na tabela de vendas e buscar a descrição daquele código em outra tabela. Para fazer isto todas as tabelas envolvidas devem ser abertas, assim como seus índices, pois será através do índice que a procura será feita.

Exemplificando através da tabela de mercadoria:

Como na tabela de vendas existe o código da mercadoria que foi vendida, e na tabela de mercadoria existe a descrição da mercadoria para aquele código, usamos o índice para procurar dentro da tabela de mercadoria, o código existente dentro da tabela de vendas. Isto é possível desde que a tabela de mercadoria esteja indexada pelo código, pois a procura será feita por ele.

Para podermos então codificarmos o formulário que terá acesso aos dados da tabela de vendas, vamos criar as variáveis que serão usadas para lidar com o banco de dados e as tabelas:

```

Option Explicit

Dim BancoDeDados As Database
Dim TBVendas As Recordset
Dim TBMercadoria As Recordset
Dim TBCliente As Recordset

```

Criamos uma variável para cada tabela que será aberta. Precisamos agora criar um evento Form_Load para abrir todas essas tabelas.

```
Project1 - frmLançamentoVendas (Code)
Form KeyPress

Private Sub Form_Load()
Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")
Set TBMercadoria = BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)
Set TBCliente = BancoDeDados.OpenRecordset("Cliente", dbOpenTable)
Set TBVendas = BancoDeDados.OpenRecordset("Vendas", dbOpenTable)

TBMercadoria.Index = "IndCódigo"
TBCliente.Index = "IndCódigo"
TBVendas.Index = "IndClienteMercadoria"
End Sub
```

Veja que abrimos todas as três tabelas que serão usadas neste formulário, e abrimos também todos os índices relativo a cada uma.

As nossas funções auxiliares sofrerão alguma mudança pois acrescentamos em nosso formulário dois **labels**, e um conterà a descrição da mercadoria e outro o nome do cliente.

```
Project1 - frmLançamentoVendas (Code)
Form KeyPress

Private Function AtualizaCampos()

TBVendas("Código") = txtCódigo
TBVendas("CódCliente") = txtCódCliente
TBVendas("ValorVenda") = txtValorVenda
TBVendas("QuantidadeVendida") = txtQuantidadeVendida

End Function
```

```
Project1 - frmLançamentoVendas (Code)
Form KeyPress

Private Function AtualizaFormulário()

txtCódigo = TBVendas("Código")
txtCódCliente = TBVendas("CódCliente")
txtValorVenda = Format(TBVendas("ValorVenda"), "Standard")
txtQuantidadeVendida = TBVendas("QuantidadeVendida")

TBMercadoria.Seek "=", txtCódigo
TBCliente.Seek "=", txtCódCliente

lblDescrição = TBMercadoria("Descrição")
lblNomeCliente = TBCliente("Nome")

End Function
```

Note que quando formos atualizar o formulário será necessário apresentar nos dois **labels** os dados contido em outras tabelas, por isto é necessário colocar um **Seek** para cada item a ser procurado. Quando o usuário, por exemplo, aperta o botão "Avançar" o programa avança um registro na tabela de Vendas e usa os códigos contidos nela para tentar localizar suas respectivas descrições e nome do cliente.

```
Project1 - frmLançamentoVendas [Code]
Form KeyPress
Private Function LimpaFormulário()
    txtCódigo = ""
    txtCódCliente = ""
    txtValorVenda = ""
    txtQuantidadeVendida = ""

    lblDescrição.Caption = ""
    lblNomeCliente.Caption = ""
End Function
```

```
Project1 - frmLançamentoVendas [Code]
(General) AtualizaFormulário
Private Function CancelaDigitação()

    AtualizaFormulário
    cmdIncluir.Enabled = True
    cmdAlterar.Enabled = True
    cmdConsultar.Enabled = True
    cmdExcluir.Enabled = True
    cmdAnterior.Enabled = True
    cmdProximo.Enabled = True
    cmdGravar.Enabled = False
    cmdSair.Enabled = True
    Frame1.Enabled = False

End Function
```

Para facilitar nosso trabalho, criamos aqui uma função **CancelaDigitação** que tem por finalidade anular o que o usuário estiver fazendo, voltar os botões e **frame** para seu estado natural. Usaremos esta função quando o usuário digitar algum código inválido.

Com as funções prontas, vamos completar a codificação do **Form_Load**, pois apenas abrimos as tabelas, precisamos agora verificar se a tabela de vendas está vazia, e desabilita o botão "Gravar" e o **frame**:

```
Project1 - frmLançamentoVendas (Code)
Form KeyPress

Private Sub Form_Load()
Set BancoDeDados = OpenDatabase(App.Path & "\Estoque.MDB")
Set TBMercadoria = BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)
Set TBCliente = BancoDeDados.OpenRecordset("Cliente", dbOpenTable)
Set TBVendas = BancoDeDados.OpenRecordset("Vendas", dbOpenTable)

TBMercadoria.Index = "IndCódigo"
TBCliente.Index = "IndCódigo"
TBVendas.Index = "indClienteMercadoria"

cmdGravar.Enabled = False
Frame1.Enabled = False

If TBVendas.EOF = False Then
AtualizaFormulário
End If

End Sub
```

A inclusão para esta tabela segue o estilo que usamos para os outros formulários. Existe somente uma diferença fundamental que é o aparecimento do nome do cliente e a descrição da mercadoria quando o usuário digitar o código correspondente. Ou seja, numa inclusão, quando o usuário digitar o código da mercadoria que foi vendida, o programa terá que acessar a tabela de Mercadoria, procurar o código que o usuário acabou de digitar, e trazer de lá a descrição daquele código.

Este trabalho todo podemos fazer no evento **LostFocus** da caixa de texto. Por que este evento? Porque esta procura será feita DEPOIS que o usuário digitar o código e passar para a próxima caixa de texto (ou seja, quando o objeto perder o foco).

Então, diante disto, precisamos criar dois evento **LostFocus**. Uma para a caixa de texto txtCódigo e outro para txtCódCliente:

```
Project1 - frmLançamentoVendas (Code)
txtCódigo LostFocus

Private Sub txtCódigo_LostFocus()
txtCódigo.Text = Format(txtCódigo.Text, "000")

TBMercadoria.Seek "=", txtCódigo

If TBMercadoria.NoMatch = True Then
MsgBox "Mercadoria não cadastrada. Operação Cancelada"
CancelaDigitação
Exit Sub
End If

lblDescrição = TBMercadoria("Descrição")

End Sub
```

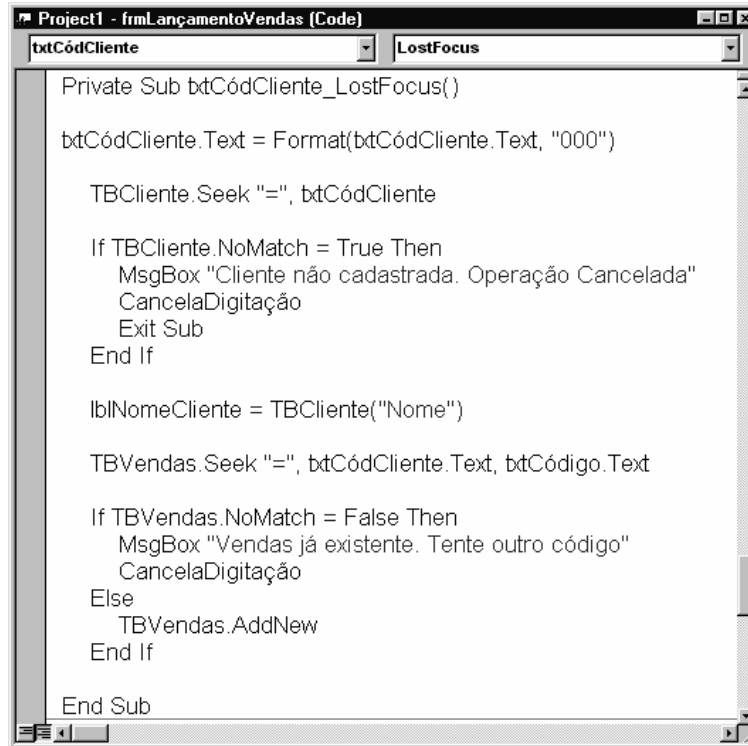
Usamos o **SEEK** para efetuar uma procura dentro da tabela TBMercadoria. O que irá procurar? Irá procurar o código digitado na caixa de texto txtCódigo.

Se **NoMatch** for igual a True é sinal que não existe nenhuma mercadoria cadastrada com o código digitado. Então uma mensagem será mostrada na tela: “Mercadoria não cadastrada”.

Depois chamamos aquela função que criamos chamada Cancela Digitação. Logicamente que se quisermos incrementar ainda mais nosso programa poderíamos fazer com que o programa abrisse uma tela mostrando quais as mercadorias que existem cadastradas para o usuário escolher uma, ou algo assim. Mas para resumir nosso exemplo deixaremos assim.

Se o código digitado na caixa de texto for encontrado então a descrição é inserida dentro do label "lblDescrição" para que seja mostrada no formulário.

Fazemos a mesma coisa com a evento **LostFocus** de txtCódCliente:



```
Project1 - frmLançamentoVendas (Code)
txtCódCliente LostFocus

Private Sub txtCódCliente_LostFocus()

    txtCódCliente.Text = Format(txtCódCliente.Text, "000")

    TBCliente.Seek "=", txtCódCliente

    If TBCliente.NoMatch = True Then
        MsgBox "Cliente não cadastrada. Operação Cancelada"
        CancelaDigitação
        Exit Sub
    End If

    lblNomeCliente = TBCliente("Nome")

    TBVendas.Seek "=", txtCódCliente.Text, txtCódigo.Text

    If TBVendas.NoMatch = False Then
        MsgBox "Vendas já existente. Tente outro código"
        CancelaDigitação
    Else
        TBVendas.AddNew
    End If

End Sub
```

A inclusão de registros dentro da tabela de vendas é igual ao apresentado nos outros programas:

```
Project1 - frmLançamentoVendas (Code)
cmdIncluir Click
Private Sub cmdIncluir_Click()
    cmdIncluir.Enabled = False
    cmdAlterar.Enabled = False
    cmdConsultar.Enabled = False
    cmdExcluir.Enabled = False
    cmdAnterior.Enabled = False
    cmdProximo.Enabled = False
    cmdGravar.Enabled = True
    cmdSair.Enabled = True

    LimpaFormulário
    Frame1.Enabled = True
    txtCódigo.SetFocus
End Sub
```

A alteração segue os padrões anteriores, lembrando que nesta tabela de vendas não podemos alterar o Código da Mercadoria e o Código do Cliente, pois ambos fazem parte da chave do índice da tabela. Se necessitar alterar o código, exclua e inclua novamente:

```
Project1 - frmLançamentoVendas (Code)
cmdAlterar Click
Private Sub cmdAlterar_Click()
    cmdIncluir.Enabled = False
    cmdAlterar.Enabled = False
    cmdConsultar.Enabled = False
    cmdExcluir.Enabled = False
    cmdAnterior.Enabled = False
    cmdProximo.Enabled = False
    cmdGravar.Enabled = True
    cmdSair.Enabled = True

    Frame1.Enabled = True
    txtCódigo.Enabled = False
    txtCódCliente.Enabled = False
    txtQuantidadeVendida.SetFocus
    TBVendas.Edit
End Sub
```

```
Project1 - frmLançamentoVendas (Code)
cmdGravar Click
Private Sub cmdGravar_Click()
|
cmdIncluir.Enabled = True
cmdAlterar.Enabled = True
cmdConsultar.Enabled = True
cmdExcluir.Enabled = True
cmdAnterior.Enabled = True
cmdProximo.Enabled = True
cmdGravar.Enabled = False
cmdSair.Enabled = True

Frame1.Enabled = False
btCódigo.Enabled = True
btCódCliente.Enabled = True
AtualizaCampos
TBVendas.Update
End Sub
```

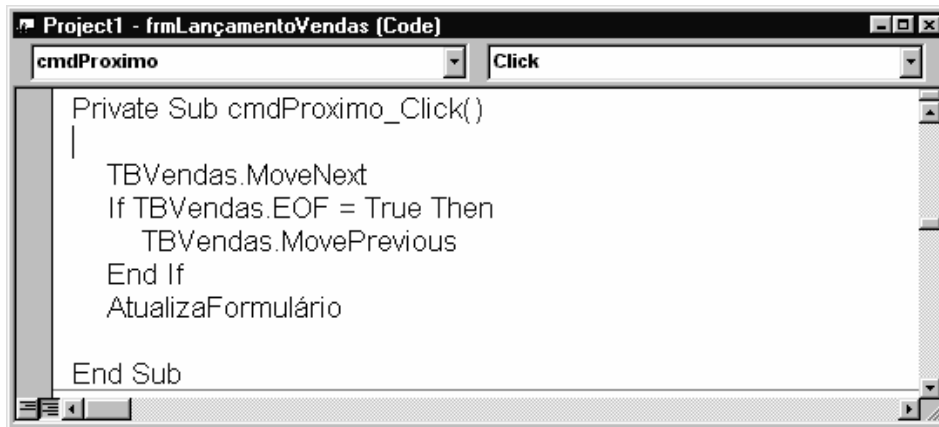
A exclusão também é semelhante aos anteriores:

```
Project1 - frmLançamentoVendas (Code)
cmdExcluir Click
Private Sub cmdExcluir_Click()
|
If MsgBox("Confirma Exclusão?", vbYesNo) = vbYes Then
    TBVendas.Delete
    cmdAnterior_Click
End If
End Sub
End Sub
```

Os botões Anterior e Próximo devem fazer uma pesquisa na tabela de Mercadoria e Clientes para atualizar os **labels**, a fim de que apareça a descrição da mercadoria e o nome do cliente:

```
Project1 - frmLançamentoVendas (Code)
cmdAnterior Click
Private Sub cmdAnterior_Click()
    TBVendas.MovePrevious
    If TBVendas.BOF = True Then
        TBVendas.MoveNext
    End If
    AtualizaFormulário
End Sub
End Sub
```

Perceba que esta rotina move a tabela de vendas para o registro anterior, e depois ao chamar a função **AtualizaFormulário** faz uma procura dentro da tabela de mercadoria e depois dentro da tabela de clientes usando como chave de busca o conteúdo existente dentro da tabela de vendas.



```
Project1 - frmLançamentoVendas (Code)
cmdProximo Click
Private Sub cmdProximo_Click()
    TBVendas.MoveNext
    If TBVendas.EOF = True Then
        TBVendas.MovePrevious
    End If
    AtualizaFormulário
End Sub
```

Dê uma conferida em seu código fonte agora:

Option Explicit

Dim BancoDeDados As Database
Dim TBVendas As Recordset
Dim TBMercadoria As Recordset
Dim TBCliente As Recordset

Private Sub cmdAlterar_Click()

cmdIncluir.Enabled = False
cmdAlterar.Enabled = False
cmdConsultar.Enabled = False
cmdExcluir.Enabled = False
cmdAnterior.Enabled = False
cmdProximo.Enabled = False
cmdGravar.Enabled = True
cmdSair.Enabled = True

Frame1.Enabled = True
txtCódigo.Enabled = False
txtCódCliente.Enabled = False
txtQuantidadeVendida.SetFocus
TBVendas.Edit

End Sub

Private Sub cmdAnterior_Click()

TBVendas.MovePrevious
If TBVendas.BOF = True Then
TBVendas.MoveNext
End If
AtualizaFormulário

End Sub

Private Sub cmdExcluir_Click()

If MsgBox("Confirma Exclusão?", vbYesNo) = vbYes Then
TBVendas.Delete
cmdAnterior_Click
End If

End Sub

Private Sub cmdGravar_Click()

```
cmdIncluir.Enabled = True
cmdAlterar.Enabled = True
cmdConsultar.Enabled = True
cmdExcluir.Enabled = True
cmdAnterior.Enabled = True
cmdProximo.Enabled = True
cmdGravar.Enabled = False
cmdSair.Enabled = True
```

```
Frame1.Enabled = False
txtCódigo.Enabled = True
txtCódCliente.Enabled = True
AtualizaCampos
TBVendas.Update
```

```
End Sub
```

```
Private Sub cmdIncluir_Click()
```

```
cmdIncluir.Enabled = False
cmdAlterar.Enabled = False
cmdConsultar.Enabled = False
cmdExcluir.Enabled = False
cmdAnterior.Enabled = False
cmdProximo.Enabled = False
cmdGravar.Enabled = True
cmdSair.Enabled = True
```

```
LimpaFormulário
Frame1.Enabled = True
txtCódigo.SetFocus
```

```
End Sub
```

```
Private Sub cmdProximo_Click()
```

```
    TBVendas.MoveNext
    If TBVendas.EOF = True Then
        TBVendas.MovePrevious
    End If
    AtualizaFormulário
```

```
End Sub
```

```

Private Sub cmdSair_Click()
    Unload frmLançamentoVendas
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)

If KeyAscii = vbKeyReturn Then
    SendKeys (" {TAB} ")
    KeyAscii = 0
End If

End Sub

Private Sub Form_Load()
Set BancoDeDados=OpenDatabase(App.Path & "\Estoque.MDB")
SetTBMercadoria=BancoDeDados.OpenRecordset("Mercadoria", dbOpenTable)
Set TBCliente=BancoDeDados.OpenRecordset("Cliente", dbOpenTable)
Set TBVendas=BancoDeDados.OpenRecordset("Vendas", dbOpenTable)

TBMercadoria.Index = "IndCódigo"
TBCliente.Index = "IndCódigo"
TBVendas.Index = "indClienteMercadoria"

cmdGravar.Enabled = False
Frame1.Enabled = False

If TBVendas.EOF = False Then
    AtualizaFormulário
End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

    TBVendas.Close
    TBMercadoria.Clone
    TBCliente.Close
    BancoDeDados.Close

End Sub

Private Function AtualizaCampos()

```

```
TBVendas("Código") = txtCódigo
TBVendas("CódCliente") = txtCódCliente
TBVendas("ValorVenda") = txtValorVenda
TBVendas("QuantidadeVendida") = txtQuantidadeVendida
```

```
End Function
```

```
Private Function LimpaFormulário()
```

```
txtCódigo = ""
txtCódCliente = ""
txtValorVenda = ""
txtQuantidadeVendida = ""
```

```
lblDescrição.Caption = ""
lblNomeCliente.Caption = ""
```

```
End Function
```

```
Private Sub txtCódigo_LostFocus()
```

```
txtCódigo.Text = Format(txtCódigo.Text, "000")
```

```
TBMercadoria.Seek "=", txtCódigo
```

```
If TBMercadoria.NoMatch = True Then
```

```
MsgBox "Mercadoria não cadastrada. Operação Cancelada"
```

```
CancelaDigitação
```

```
Exit Sub
```

```
End If
```

```
lblDescrição = TBMercadoria("Descrição")
```

```
End Sub
```

```
Private Sub txtCódCliente_LostFocus()
```

```
txtCódCliente.Text = Format(txtCódCliente.Text, "000")
```

```
TBCliente.Seek "=", txtCódCliente
```

```
If TBCliente.NoMatch = True Then
```

```
MsgBox "Cliente não cadastrada. Operação Cancelada"
```

```
CancelaDigitação
```

```
Exit Sub
```



```

End If

lblNomeCliente = TBCliente("Nome")

TBVendas.Seek "=", txtCódCliente.Text, txtCódigo.Text

If TBVendas.NoMatch = False Then
    MsgBox "Vendas já existente. Tente outro código"
    CancelaDigitação
Else
    TBVendas.AddNew
End If

End Sub

Private Sub txtValorVenda_LostFocus()
txtValorVenda.Text = Format(txtValorVenda.Text, "Standard")
End Sub

Private Function AtualizaFormulário()

    txtCódigo = TBVendas("Código")
    txtCódCliente = TBVendas("CódCliente")
    txtValorVenda = Format(TBVendas("ValorVenda"), "Standard")
    txtQuantidadeVendida = TBVendas("QuantidadeVendida")
    TBMercadoria.Seek "=", txtCódigo
    TBCliente.Seek "=", txtCódCliente
    lblDescrição = TBMercadoria("Descrição")
    lblNomeCliente = TBCliente("Nome")

End Function

Private Function CancelaDigitação()
    AtualizaFormulário
    cmdIncluir.Enabled = True
    cmdAlterar.Enabled = True
    cmdConsultar.Enabled = True
    cmdExcluir.Enabled = True
    cmdAnterior.Enabled = True
    cmdProximo.Enabled = True
    cmdGravar.Enabled = False
    cmdSair.Enabled = True
    Frame1.Enabled = False
End Function

```

Deixamos o botão Consulta de fora propositadamente. Vamos voltar nele nos próximos capítulos.

