Excel

51 MACROS Incríveis VBA

Rotinas Prontas Para Usar e Facilitar a Sua Vida

Sistema de login e senha planilha

Procurar erros automáticamente

Salvar em diferentes diretórios

Enviar e-mail automático

Troca rápida de vínculo

E muito mais...

PREMIUM
Good Book
BEST PRICE
Awesome

Luiz Felipe G. Araujo

Sumário

OCULTAR E MOSTRAR INFORMAÇÕES

OUTROS

INTERAÇÃO COM WINDOWS

INTERAÇÕES COM OUTOLOOK

Introdução

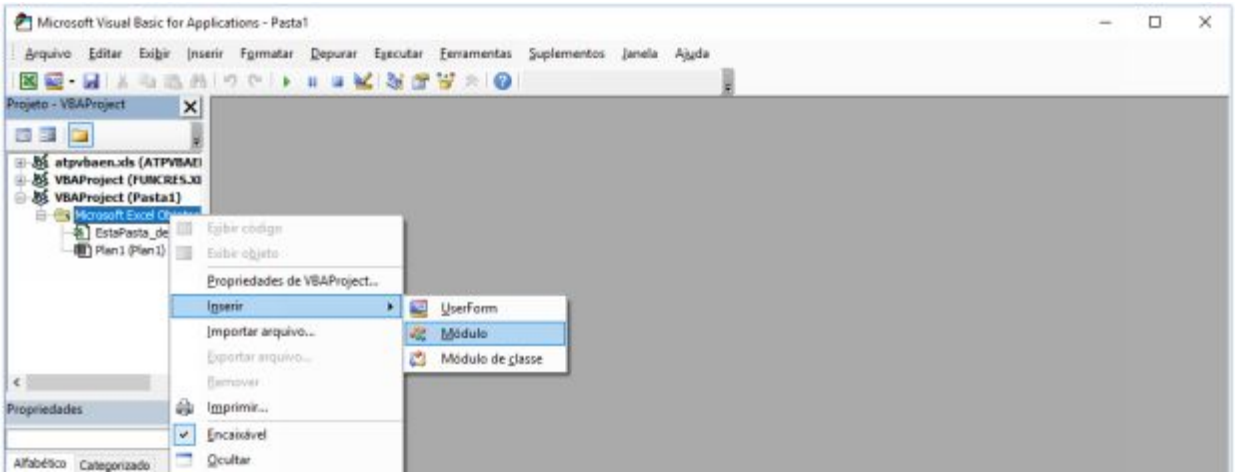
Este livro é uma solução rápida para usuários que trabalham com planejamento, finanças, comercial, logística e todas as demais áreas que lidam com grandes quantidades de informações. É importante analisar as necessidades de cada projeto, processo ou sistema, para aplicar corretamente a rotinas desejadas.

Esta breve introdução se destina aos usuários que não sabem o procedimento inicial básico de implementação de uma macro, serão instruções rápidas e diretas para demonstrar a implementação básica dos códigos.

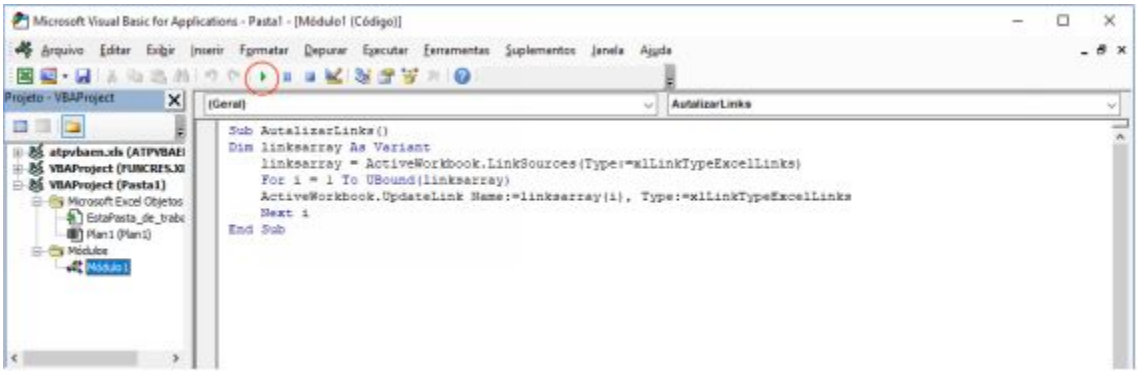
São apenas 3 passos para adicionar suas macros:

1. Acessar o menu desenvolvedor, clicar no botão Visual Basic (Ou atalho Alt + F11).

2. Escolher o objeto no qual a macro será inserida, de maneira prática, na grande maioria dos casos, poderá se adicionar as macros em um novo módulo, clicando com botão direito na pasta dos objetos, Inserir, Módulo.

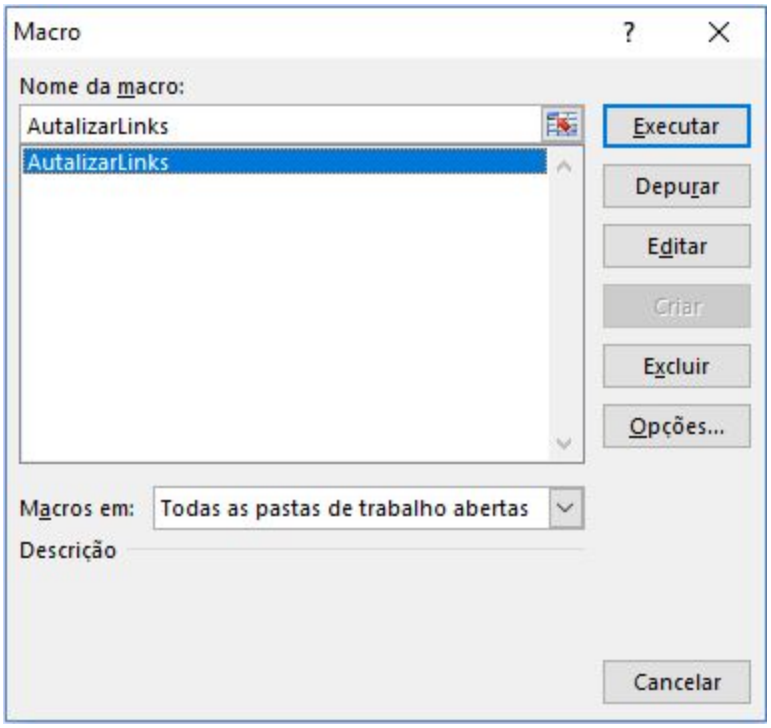


3. Inserir a macro desejada na caixa de texto e testar seu funcionamento clicando no botão “Play” ou pressionando F5. É possível também executar um “Passo a Passo”, da macro, para verificar o que cada linha de código executa, com o atalho F8, para depuração.



Obs 1: Para algumas macros, será necessário inserir as macros no objeto “EstaPasta” e em outros casos, na planilha desejada, por exemplo: “Plan1”.

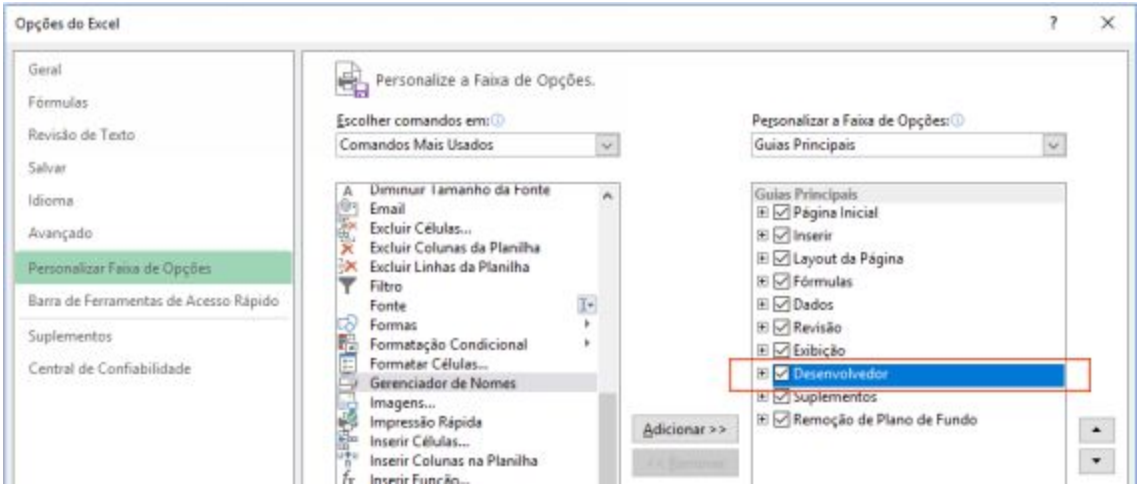
Obs 2: Uma vez que a macro já esteja inserida, a janela do Visual Basic, pode ser fechada, a mesma poderá ser acessada de maneira mais prática, através da Guia Desenvolvedor, botão Macros.



Obs 3: Caso a guia “Desenvolvedor” não esteja disponível, a mesma poderá ser adicionada da seguinte maneira:

Arquivo – Opções – Personalizar Faixa de Opções

Na seleção, basta marcar a opção desenvolvedor.



Abrir Salvar e Fechar

1. Abrir um arquivo, caso já esteja aberto, maximiza-lo.

Uma macro muito utilizada para acelerar processos é a de abrir arquivos, porém por muitas vezes o arquivo já está aberto por alguma verificação manual do usuário, esta macro faz uma verificação, caso o arquivo já esteja aberto, ele é maximizado.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub AbrirMaximizar()
Dim Arquivo, Diretorio, Extensao As String
Diretorio = " C:\Users\usuarioteste\Documents "
Arquivo = " Planilha1 "
Extensao = " xlsx "
Dim wkb As Workbook
On Error Resume Next
Set wkb = application.Workbooks(Arquivo)
If Err <> 0 Then
Set wkb = application.Workbooks.Open(Diretorio + "\" +
Arquivo + "." & Extensao, UpdateLinks:=0)
End If
On Error GoTo 0
Workbooks(Arquivo).Activate
End Sub
```

2. Salvar automaticamente antes de fechar

Esta macro é extremamente simples, porém muito útil, com esta rotina o Excel sempre salvará o arquivo quando for fechado, a rotina é útil para agilizar o processo de salvar e também previne o fechamento acidental sem salvar.

Obs: É necessário incluir a macro dentro do objeto WorkBook.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
Application.DisplayAlerts = False
ActiveWorkbook.Save
Application.DisplayAlerts = True
End Sub
```

3. Copiar uma aba, converter para valor e salvar saída

Uma rotina muito comum e de grande utilidade é a de “gerar saída” de dados. Visto que muitos relatórios tem apenas uma aba de informação a ser divulgada, desta forma, esta aba será copiada, terá seus dados convertidos em valor e será salvada em uma pasta destino escolhida.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub CopiarSaida()
Dim Caminho, NomeArquivo, AbaSaida As String
AbaSaida = "Plan3"
Caminho = "C:\Users\UsuarioTeste\Documents\"
NomeArquivo = "Planilha.xls"
Worksheets("Plan3").Copy
Cells.Copy
Cells.PasteSpecial Paste:=xlPasteValues
ActiveWorkbook.SaveAs Filename:=Caminho &
NomeArquivo, _
FileFormat:=xlOpenXMLWorkbookMacroEnabled
End Sub
```

4. Salvar backup rápido

Esta macro tem o objetivo de salvar rapidamente uma cópia de segurança do arquivo, na mesma pasta de origem com data e hora.

```
Sub SalvarBackUp()
ThisWorkbook.SaveCopyAs Filename:=ThisWorkbook.Path &
_
"\" & Format(Date, "mm-dd-yy") & " " & _
ThisWorkbook.Name
End Sub
```

5. Salvar uma cópia backup ao fechar o arquivo

Esta macro, automaticamente salva uma cópia do arquivo antes de fechar. Esta cópia recebe o horário e data do momento em que o arquivo foi salvo.

Obs: É necessário incluir a macro dentro do objeto WorkBook.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
Application.DisplayAlerts = False
FileDefaultName = "TestFile"
Application.DisplayAlerts = False
ActiveWorkbook.SaveAs
Filename:=Application.ActiveWorkbook.Path & "\" &
Format(Time, "hhmmss") & " " & Format(Date, "mm-dd-yy")
& " " & FileDefaultName
Application.DisplayAlerts = True
End Sub
```

6. Salvar cada planilha como arquivo Excel

Esta rotina tem por objetivo salvar cada aba como um arquivo diferente em um único destino.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub SalvarAbas()
Dim wkb, NomeArquivo As String
wkb = ActiveWorkbook.Name
Diretorio = "C:\Users\usuarioteste\Documents"
For i = 1 To Worksheets.Count
Worksheets(i).Select
NomeArquivo = ActiveSheet.Name
ActiveSheet.Copy
ActiveWorkbook.SaveAs Filename:=Diretorio + "\" +
NomeArquivo + "." & "xlsx"
ActiveWorkbook.Close
Workbooks(wkb).Activate
Next
MsgBox ("Arquivos salvados com sucesso")
End Sub
```

7. Salvar cada aba como um arquivo PDF

Esta rotina tem por objetivo salvar cada aba como um arquivo diferente em um único destino no formato PDF.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub SaveWorkshetAsPDF()
Dim ws As Worksheet
Diretorio = "C:\Users\usuarioteste\Documents"
For Each ws In Worksheets
On Error Resume Next
ws.ExportAsFixedFormat xlTypePDF, Diretorio & "\" &
ws.Name & ".pdf"
Next ws
End Sub
```

3. Salvar o arquivo em diferentes pastas

É comum distribuir o arquivo na rede em diversos locais, para tanto, esta macro permite ao usuário salvar o arquivo em diversas pastas de maneira rápida e prática.

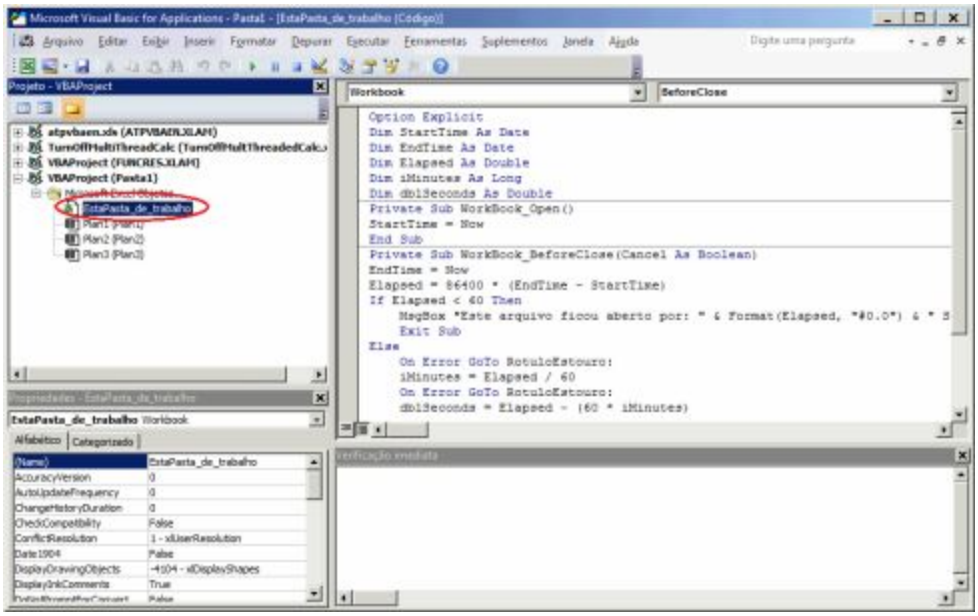
Obs1: É preciso alterar o nome do arquivo, a extensão e o diretório, marcados em negrito.
Obs2: Para alterar, apagar ou incluir novos diretórios, basta manipular os já exemplificados abaixo. Para adicionar, basta criar Diretorio(4), Diretorio(5) e assim seucessivamente.

```
Sub SalvarArquivo()
Dim NomeArquivo, Extensao As String
Dim Diretorio(1 To 999) As String
NomeArquivo = " NomeDoArquivo "
Extensao = " xls "
Diretorio(1) = " C:\Users\usuarioteste\Documents "
Diretorio(2) = " C:\Users\usuarioteste\Documents "
Diretorio(3) = " C:\Users\usuarioteste\Documents "
For i = 1 To 999
    If Diretorio(i) = "" Then
        GoTo RotuloSaida:
    End If
    Perg = MsgBox("Deseja salvar com o nome: " &
NomeArquivo & " no caminho: " & Diretorio(i), vbYesNo,
"Salvar Arquivo")
    If Perg = vbYes Then
        ActiveWorkbook.SaveAs Filename:=Diretorio(i) + "\" +
NomeArquivo + "." & Extensao
    End If
Next i
RotuloSaida:
MsgBox "Arquivos salvados com sucesso"
End Sub
```

9. Quando fechar o arquivo informar o tempo gasto

Esta é uma macro muito comum para o controle de rotina do usuário, onde o mesmo pode verificar o seu rendimento durante a utilização de uma planilha.

Obs1: É necessário incluir a macro dentro do objeto WorkBook.



Option Explicit
Dim StartTime As Date
Dim EndTime As Date
Dim Elapsed As Double
Dim iMinutes As Long
Dim dblSeconds As Double
Private Sub WorkBook_Open()
StartTime = Now
End Sub
Private Sub WorkBook_BeforeClose(Cancel As Boolean)
EndTime = Now
Elapsed = 86400 * (EndTime - StartTime)
If Elapsed < 60 Then

```
    MsgBox "Este arquivo ficou aberto por: " &
Format(Elapsed, "#0.0") & " Segundos", vbInformation +
vbOKOnly
    Exit Sub
Else
    On Error GoTo RotuloEstouro:
    iMinutes = Elapsed / 60
    On Error GoTo RotuloEstouro:
    dblSeconds = Elapsed - (60 * iMinutes)
    MsgBox "Este arquivo ficou aberto por: " &
Format(iMinutes, "#") & ":" & Format(dblSeconds, "00") & "
Minutos", vbInformation + vbOKOnly
    Exit Sub
End If
RotuloEstouro:
MsgBox "O tempo será computado a partir do próximo
acesso"
End Sub
```

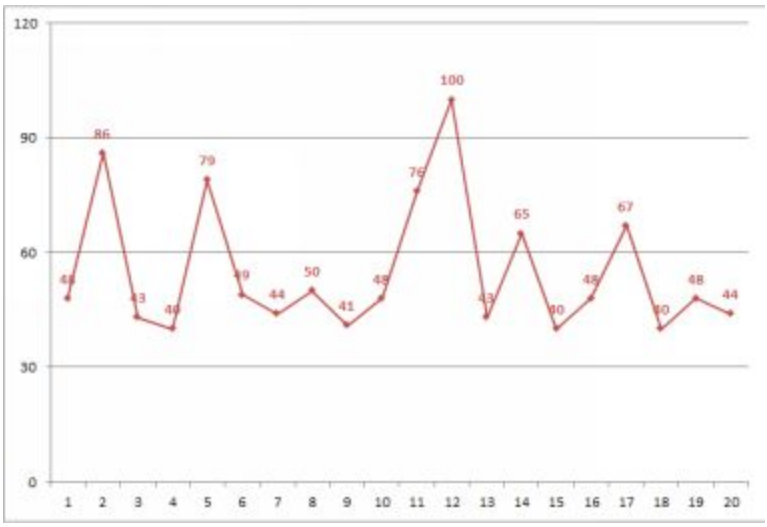
Interação com gráficos

0. Automaticamente ajustar os rótulos dos gráficos

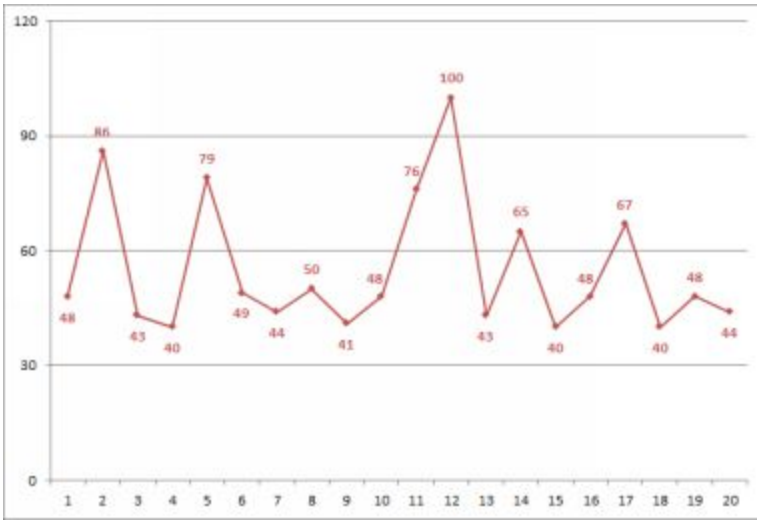
Este código automaticamente ajusta os rótulos, considerando o crescimento ou decrescimento do gráfico, evitando desta maneira a colisão do rótulo com o gráfico, conforme as imagens abaixo:

Obs: O código ajustará todos os gráficos da planilha ativa

Before:



After:




```
Sub AjustGraphic()
For i = 1 To ActiveSheet.ChartObjects.Count
ActiveSheet.ChartObjects(i).Select
Dim MaxScale, MinScale, MyPoint, DefaultPosition,
AdjustedPosition As Long
Dim MySeries As Series
Dim PointsArray As Variant
With ActiveChart
    MaxScale = .Axes(xlValue).MaximumScale
    MinScale = .Axes(xlValue).MinimumScale
    For Each MySeries In .SeriesCollection
```

```
        If MySeries.ChartType <> xlColumnClustered And _
           MySeries.ChartType <> xlLine And _
           MySeries.ChartType <> xlLineMarkers Then
           GoTo NEXTSERIES
        End If
           PointsArray = MySeries.Values
        For MyPoint = LBound(PointsArray) To
UBound(PointsArray)
            If MySeries.Points(MyPoint).HasDataLabel = False
Then
            GoTo NEXTDOT
            End If
            If MySeries.ChartType = xlColumnClustered Then
               MySeries.Points(MyPoint).DataLabel.Position =
xlLabelPositionOutsideEnd
               If PointsArray(MyPoint) > MaxScale * 0.9 Then
                  MySeries.Points(MyPoint).DataLabel.Position
= xlLabelPositionInsideEnd
               End If
            End If
            If MySeries.ChartType = xlLine Or
MySeries.ChartType = xlLineMarkers Then
               MySeries.Points(MyPoint).DataLabel.Position =
xlBelow
               If MyPoint > 1 Then
               If PointsArray(MyPoint) > PointsArray(MyPoint -
1) Then
                  MySeries.Points(MyPoint).DataLabel.Position
= xlAbove
               Else
                  MySeries.Points(MyPoint).DataLabel.Position
= xlBelow
               End If
            End If
            If PointsArray(MyPoint) > MaxScale * 0.9 Or _
               PointsArray(MyPoint) < MinScale * 1.5 Then
```

```
                MySeries.Points(MyPoint).DataLabel.Position =
xlRight
            End If
        End If
NEXTDOT:
    Next MyPoint
NEXTSERIES:
  Next MySeries
End With
Next
End Sub
```

1. Redimensionar todos os gráficos

Padronizar todos os gráficos de um arquivo pode ser uma tarefa árdua, esta macro facilita este processo ao estabelecer as mesmas dimensões, para tanto, basta modificar a largura e a altura destacados em negrito.

Obs 1: Substituir os valores de exemplo em negrito pelos valores desejados.
Obs 2: Widht será a largura e Height será a altura de todos os gráficos

```
Sub RedmGraf()
Dim i As Integer
For i = 1 To ActiveSheet.ChartObjects.Count
With ActiveSheet.ChartObjects(i)
.Width = 300
.Height = 200
End With
Next i
End Sub
```

Vínculos com Arquivos Externos

2. Atualizar todos os links

Esta macro é essencial para usuários que lidam com grandes quantidades de informação e os vínculos não são atualizados automaticamente. Com esta rotina, todos serão atualizados automaticamente ao executa-la.

```
Sub AutalizarLinks()
Dim linksarray As Variant
    linksarray =
ActiveWorkbook.LinkSources(Type:=xlLinkTypeExcelLinks)
    For i = 1 To UBound(linksarray)
    ActiveWorkbook.UpdateLink Name:=linksarray(i),
Type:=xlLinkTypeExcelLinks
    Next i
End Sub
```

3. Quebrar todos os vínculos

Esta rotina tem por objetivo quebrar todos os vínculos de um arquivo.

```
Sub QuebrarVinculos()
Dim alinksarray As Variant
    alinksarray =
ActiveWorkbook.LinkSources(Type:=xlLinkTypeExcelLinks)
    Do Until IsEmpty(alinksarray)
        ActiveWorkbook.BreakLink Name:=alinksarray(1),
Type:=xlLinkTypeExcelLinks
        alinksarray =
ActiveWorkbook.LinkSources(Type:=xlLinkTypeExcelLinks)
    Loop
```

```
End Sub
```

4. Alterar vínculo

Da mesma maneira que a macro anterior, a alteração de vínculos se dá por um processo manual e lento, desta forma, através desta rotina, é possível executar esta tarefa de maneira rápida e automatizada.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados. Arquivo1 é o vínculo atual, Arquivo2 é o vínculo para o qual será alterado.

```
Sub TrocarVinculo()
Dim Arquivo1, Arquivo2 As String
Dim Cont As Integer
Dim wkb, wkb2 As String
wkb = ActiveWorkbook.Name
Arquivo1 = " C:\Users\usuarioteste\Documents\
Planilha1.xls "
Arquivo2 = " C:\Users\usuarioteste\Documents\
Planilha2.xls "
Application.DisplayAlerts = False
Workbooks.Open Filename:=Arquivo2, UpdateLinks:=0
wkb2 = ActiveWorkbook.Name
Workbooks(wkb).Activate
ActiveWorkbook.ChangeLink Name:=Arquivo1,
NewName:=Arquivo2, Type:=xlLinkTypeExcelLinks
Workbooks(wkb2).Activate
ActiveWorkbook.Close
Application.DisplayAlerts = False
Workbooks(wkb).Activate
End Sub
```

Controle e segurança das informações

5. Computar dados de usuários que acessaram a planilha

Para muitos setores, é essencial ter um controle dos usuários que acessaram e/ou modificaram os dados de uma planilha, desta forma, esta rotina computa o usuário, data e horário de acesso.

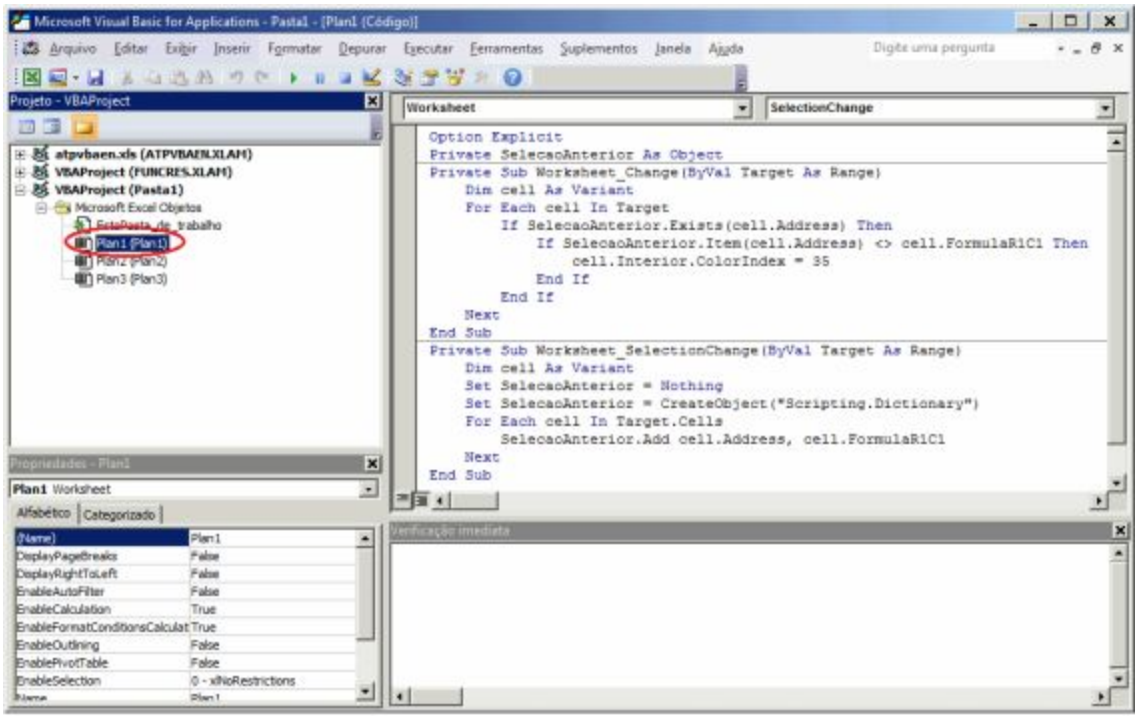
Obs: É preciso criar uma aba chamada “ **Controle de Acesso** ”, onde os dados dos usuários serão armazenados.

```
Sub Auto_Open()
Dim linhas As Integer
Dim Data, Hora As Date
Data = Date
Hora = Time
linha =
Application.WorksheetFunction.CountA(Worksheets("Control
e de Acesso").Range("A1", "A1048576")) + 1
Worksheets("Controle de Acesso").Range("A" & linha) =
Application.UserName
Worksheets("Controle de Acesso").Range("B" & linha) =
Data
Worksheets("Controle de Acesso").Range("C" & linha) =
Hora
Columns(1).AutoFit
Columns(2).AutoFit
Columns(3).AutoFit
End Sub
```

6. Marcar todas as células editadas pelo usuário

Esta é uma outra rotina que tem por objetivo, estabelecer um controle dos dados de uma planilha.

Obs1: É necessário incluir a macro dentro do objeto da planilha desejada, no exemplo abaixo a planilha que destacará as alterações será a “Plan1”.




```
Option Explicit
Private SelecaoAnterior As Object
Private Sub Worksheet_Change(ByVal Target As Range)
    Dim cell As Variant
    For Each cell In Target
        If SelecaoAnterior.Exists(cell.Address) Then
            If SelecaoAnterior.Item(cell.Address) <>
cell.FormulaR1C1 Then
                cell.Interior.ColorIndex = 35
            End If
        End If
```

7. Proteger todas as planilhas

Esta simples rotina, protege automaticamente todos os dados de todas as planilhas de possíveis alterações.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub ProtectSheets()
Dim wksht As Worksheet
For Each wksht  In ActiveWorkbook.Worksheets
wksht.Protect Password:=" Senha "
Next wsheet
End Sub
```

8. Desproteger todas as planilhas

Este código é simplesmente o contrário do anterior, ele tem a função de desproteger todas as planilhas.

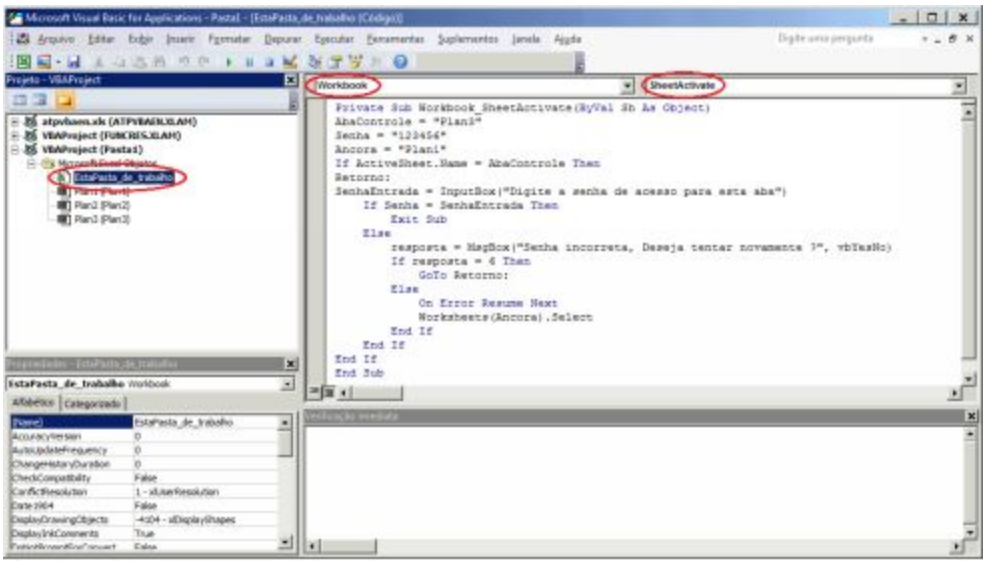
Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub UnrotectSheets()
Dim wksht As Worksheet
For Each wksht  In ActiveWorkbook.Worksheets
wksht.Unrotect Password:=" Senha "
Next wsheet
End Sub
```

9. Proteger uma aba com senha

Macro de controle de acesso para acessar uma aba específica.

Obs1: A variável **AbaControle** é a que se deseja proteger com senha.
Obs2: **Senha** é o valor que deverá ser estabelecido como regra de acesso.
Obs2: **Ancora** é a aba que será selecionada, caso o usuário erre a senha.
Obs4: O código precisa ser inserido dentro do objeto Workbook para declaração “SheetActivate”, de modo a ser executado junto com a abertura do arquivo, conforme imagem abaixo:



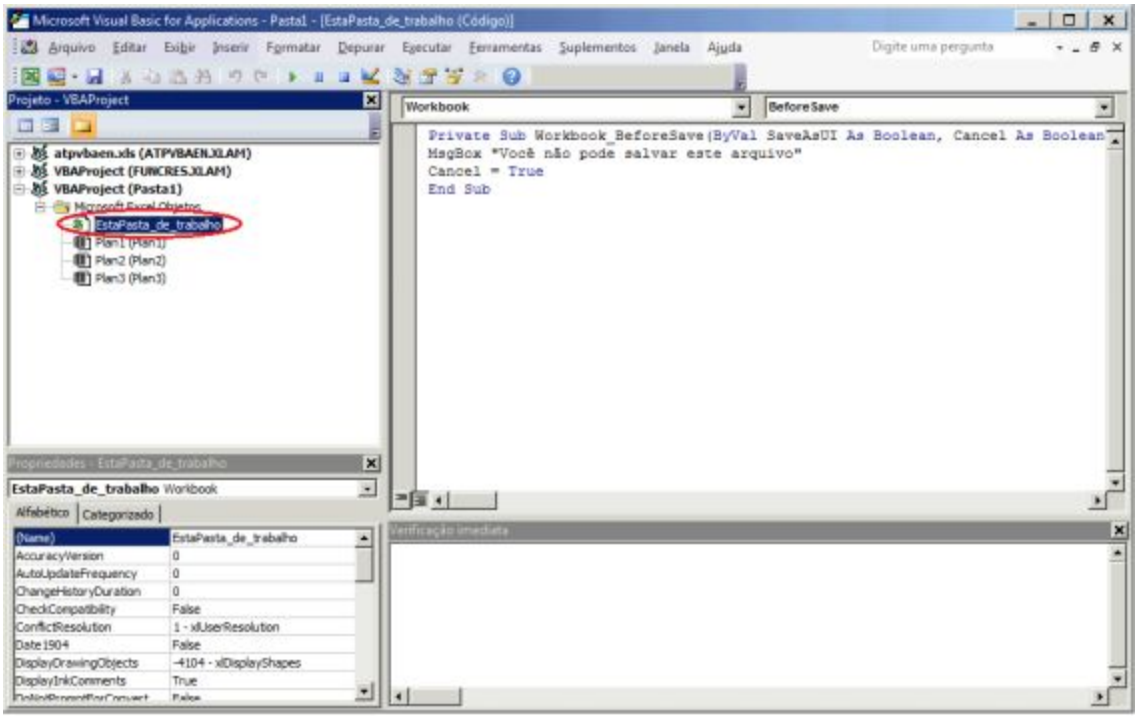
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
AbaControle = " **Plan3** "
Senha = " **123456** "
Ancora = " **Plan1** "
If ActiveSheet.Name = AbaControle Then
Retorno:
SenhaEntrada = InputBox("Digite a senha de acesso para esta aba")
    If Senha = SenhaEntrada Then
        Exit Sub
    Else

```
        resposta = MsgBox("Senha incorreta, Deseja tentar
novamente ?", vbYesNo)
        If resposta = 6 Then
            GoTo Retorno:
        Else
            On Error Resume Next
            Worksheets(Ancora).Select
        End If
    End If
End If
End Sub
```

20. Impedir o usuário salvar o arquivo

Esta macro tem por objetivo impedir o usuário de salvar o arquivo, impedindo assim a substituição de dados.

Obs1: É necessário incluir a macro dentro do objeto WorkBook.



Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)

```
Msgbox “Você não pode salvar este arquivo”
Cancel = True
End Sub
```

1. Simples Sistema de login e senha para acessar o arquivo

Este é um sistema simples de Login para apenas usuários cadastrados acessarem a planilha.

Obs1: É preciso alterar o nome do arquivo, a extensão e o diretório, marcados em negrito.

Obs2: O código precisa da declaração Auto_Open() para funcionar.

Obs3: Para alterar e criar usuários com senha, basta alterar os trechos em negrito, para adicionar, basta incluir uma nova linha, seguindo a numeração, Usuario(4), Usuario(5) e assim sucessivamente. Para senha é a mesma lógica, Senha(4), Senha(5) e assim sucessivamente.

```
Sub Auto_Open()
Dim Usuario(1 To 999) As String
Dim Senha(1 To 999) As String
Dim UsuOK As Boolean
Dim SenhaOK As Boolean
'Cadastro de Usuários
'------------------
Usuario(1) = "Luiz"
Usuario(2) = "Maria"
Usuario(3) = "Clara"
'------------------
Senha(1) = "1345"
Senha(2) = "1234"
Senha(3) = "5367"
'------------------
UsuOK = False
```

```
SenhaOK = False
Retorno:
UsuarioEntrada = InputBox("Digite o seu nome de usuário:
")
For i = 1 To 999
    If Usuario(i) = UsuarioEntrada And Usuario(i) <> "" Then
    UsuOK = True
    GoTo Rotulo1:
    End If
Next
Rotulo1:
If UsuOK = True Then
    SenhaEntrada = InputBox("Bem Vindo(a): " &
UsuarioEntrada & ", digite a sua senha:")
    If Senha(i) = SenhaEntrada Then
    Exit Sub
Else
    pergunta = MsgBox("Senha incorreta, deseja tentar
novamente ?", vbYesNo)
    If pergunta = 6 Then
        GoTo Retorno:
    Else
        MsgBox "O Arquivo será fechado"
        ActiveWorkbook.Close
    End If
End If
Else
    pergunta = MsgBox("Usuário não encontrado, deseja
tentar novamente ?", vbYesNo)
    If pergunta = 6 Then
        GoTo Retorno:
    Else
        MsgBox "O Arquivo será fechado"
        ActiveWorkbook.Close
    End If
End If
```

```
End Sub
```

Controle de erros

22. Verificar todas as abas para encontrar erros

Esta macro tem por objetivo rodar todas as abas disponíveis de um arquivo, procurando por erros comuns de formula como #N/D, #REF!, #DIV/0! e etc. O erro será localizado e informado ao usuário por meio de mensagem, informando a Aba e a Célula onde o erro se encontra.

```
Sub VerificarErros()
Dim ws As Worksheet
Dim ra As Range
For Each ws In Worksheets
    For Each ra In ws.UsedRange
        ra.Select
        On Error Resume Next
        If IsError(ra.Value) Then
           MsgBox "Aba: " & ra.Parent.Name & Chr(13) &
"Célula: " & ra.Address
        End If
    Next
Next
End Sub
```

23. Verificar uma seleção para encontrar erros

Esta macro tem a mesma mecânica da anterior, porém ao invés de verificar todas as abas para encontrar erros, a verificação é feita nas células selecionadas pelo usuário.

```
Sub VerificarErros()
Dim ra As Range
For Each ra In Selection
    ra.Select
    If IsError(ra.Value) Then
       MsgBox "Aba: " & ra.Parent.Name & Chr(13) & "Célula: "
& ra.Address
    End If
Next
End Sub
```

24. Verificar todas as abas para contabilizar erros

Esta macro tem por objetivo rodar todas as abas disponíveis de um arquivo, procurando por erros comuns de formula como #N/D, #REF!, #DIV/0! e etc. O erro será localizado e contabilizado para informar ao usuário a quantidade.

```
Sub VerificarErros()
Dim ws As Worksheet
Dim ra As Range
Dim Contador As Long
Contador = 0
For Each ws In Worksheets
    For Each ra In ws.UsedRange
        If IsError(ra.Value) Then
           Contador = Contador + 1
        End If
    Next
Next
MsgBox (Contador & " Erros encontrados")
End Sub
```

Ocultar e mostrar informações

25. Exibir todas as linhas e colunas ocultas

Ao aplicar esta macro, todas as linhas e colunas ocultas em todas as tabelas serão exibidas.

```
Sub TirarOculta()
Dim Ws As Worksheet
For Each Ws In Application.Worksheets
Ws.Select
Cells.Select
Selection.EntireRow.Hidden = False
Selection.EntireColumn.Hidden = False
Next
End Sub
```

26. Ocultar e mostrar todas as abas

Rotina utilizada para acelerar o processo de ocultar e mostrar abas. Todas as abas, exceto a primeira serão ocultas, ao rodar novamente, todas serão exibidas.

```
Sub OcultareMostrarAbas()
For i = 2 To Worksheets.Count
    If Worksheets(i).Visible = True Then
        Worksheets(i).Visible = False
    Else
        Worksheets(i).Visible = True
    End If
```

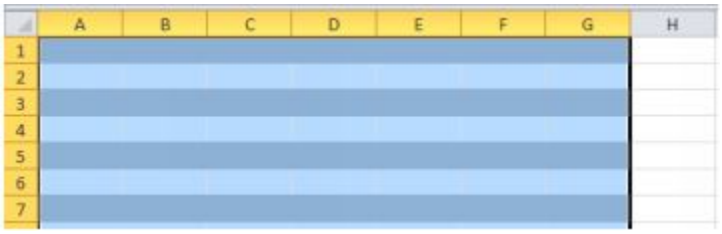
```
Next
End Sub
```

Outros

27. Aplicar cores alternadas em uma seleção

As cores intercaladas facilitam a leitura de dados, a única maneira de fazer isso no Excel sem macro, é aplicando a formatação de tabela, função nem sempre desejada. Esta macro faz esta função de forma simples e direta, com a cor desejada.

Obs: para trocar as cores basta trocar o código da cor destacado em negrito.




```
Sub CoresIntercaladas()
Dim Selecao As Range
Dim Linha As Range
Set Selecao = Selection
For Each Linha In Selecao.Rows
    If Linha.Row Mod 2 = 1 Then
        Linha.Interior.ColorIndex = 15
    End If
Next Linha
End Sub
```

28. Consolidar todos as abas na primeira

Um processo manual muito comum é a consolidação de dados, onde muitas vezes um arquivo bruto extraído do banco de dados, vem fragmentado em diversas abas. Esta rotina permite copiar dados de diversas abas e consolidar em uma primeira.

Obs1: A macro está configurada para agrupar os dados que começam a partir da célula A1.

Obs2: A primeira aba receberá as informações das demais.

```
Sub ConsolidarDados()
Dim total As Long
For i = 2 To Worksheets.Count
    Worksheets(i).Select
    Range("A1").Select
    If ActiveCell.Offset(0, 1) <> "" Then
        Range(Selection, Selection.End(xlToRight)).Select
    End If
    If ActiveCell.Offset(1, 0) <> "" Then
    Range(Selection, Selection.End(xlDown)).Select
    End If
Selection.Copy
Worksheets(1).Select
total =
Application.WorksheetFunction.CountA(Worksheets(1).Range
("A1:A1048576"))
Range("A" & total + 1).Select
ActiveSheet.Paste
Next
End Sub
```

29. Cronometrar o tempo de outras macros

Algumas macros tem processamento demorado, desta forma, é sempre recomendado cronometrar o tempo de execução para se fazer ajustes e melhorias. A função desta macro é cronometrar o tempo de execução de outras macros, desta forma, basta substituir o trecho em negrito destacado pelo código desejado.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub MacroCronometro()
'Início cronômetro
ti = Time
'Insira seu código
'Fim cronômetro
tf = Time
'Subtração para saber o tempo total de execução da macro
tDif = tf - ti
MsgBox "Tempo de processamento: " &
WorksheetFunction.text(tDif, "HH:MM:SS")
End Sub
```

30. Copar e colar valor em todas as abas

Macro utilizada comumente para gerar saída de dados para outros departamentos ou usuários.

```
Sub PassarValor()
pergunta = MsgBox("Esta macro converterá todas as células
para valores, deseja continuar ?", vbYesNo)
If pergunta = vbYes Then
    For i = 1 To Worksheets.Count
        Worksheets(i).Select
        If Worksheets(i).Visible = False Then
            Worksheets(i).Visible = True
        End If
        Cells.Select
```

```
        Selection.Copy
        Selection.PasteSpecial Paste:=xlPasteValues
        Range("A1").Select
    Next
End If
End Sub
```

1. Remover espaços em branco dentro das células

O objetivo desta macro é remover os espaços em branco dentro das células, para tanto, basta selecionar as células desejadas e rodar a macro.

```
Sub RemoverEspacos()
Dim Celula As Range
For Each Celula In Selection
If Not IsEmpty(Celula) Then
Celula = Trim(Celula)
End If
Next Celula
End Sub
```

32. Atualizar todas as tabelas dinâmicas

Esta macro é simples, mas eficaz, ela atualiza todas as tabelas dinâmicas de uma planilha.

```
Sub Refresh_PivotTables()
  Dim pivotTable As PivotTable
  For Each pivotTable In ActiveSheet.PivotTables
    pivotTable.RefreshTable
  Next
End Sub
```

33. Remover duplicatas em todas as abas

Esta é uma rotina para aplicar o tratamento de dados nas planilhas, permitindo ao usuário remover as duplicatas de todas as abas.

```
Sub RemoverDuplicatas()
coluna = InputBox("Em qual coluna deseja remover duplicata de todas as abas ?")
For i = 1 To Worksheets.Count
    Worksheets(i).Select
    ActiveSheet.Range("$" & coluna & "$1:$" & coluna & "$9999").RemoveDuplicates Columns:=1, Header:=xlNo
Next
End Sub
```

34. Consolidar dados na primeira aba

Esta macro tem por objetivo consolidar todas as abas dos arquivos abertos em um novo arquivo.

```
Sub JuntarArquivos()
Dim wkbDestino As String
Dim WorkbookName(1 To 99) As String
Dim ws As Worksheet
Dim i As Integer
i = 1
For Each Workbook In Workbooks
i = 1 + 1
WorkbookName(i) = Workbook.Name
Next Workbook
Total = i
Workbooks.Add
```

```
wkbDestino = ActiveWorkbook.Name
For i = 1 To Total
    Workbooks(i).Activate
    For Each ws In Workbooks(i).Worksheets
    ws.Copy
after:=Workbooks(wkbDestino).Sheets(Workbooks(wkbDesti
no).Sheets.Count)
    Next ws
Next i
Application.DisplayAlerts = False
For i = 1 To 3
Sheets(1).Delete
Next i
Application.DisplayAlerts = True
End Sub
```

35. Deletar abas que estão vazias

Esta macro verifica todas as planilhas para encontrar alguma com todas as células em branco, uma vez que sejam encontradas, serão deletadas.

```
Sub DeletarAbasVazias()
Dim Ws As Worksheet
On Error Resume Next
Application.DisplayAlerts = False
For Each Ws In Application.Worksheets
If Application.WorksheetFunction.CountA(Ws.UsedRange) =
0 Then
Ws.Delete
End If
Next
Application.DisplayAlerts = True
End Sub
```

36. Ordenar as abas em ordem alfabética

Esta macro, ao ser executada ordena as abas em ordem alfabética crescente, para trocar para decrescente, basta alterar o sinal de maior (“>”), destacado em negrito por menor (“<”).

```
Sub Ordenarabas()
Dim i As Integer
Dim j As Integer
Dim Resposta As VbMsgBoxResult
Resposta = MsgBox("Deseja ordenar em ordem Crescente
?", vbYesNo + vbQuestion + vbDefaultButton1, "Ordenar
Abas")
For i = 1 To Sheets.Count
For j = 1 To Sheets.Count - 1
If Resposta = vbYes Then
If UCase$(Sheets(j).Name) > UCase$(Sheets(j + 1).Name)
Then
Sheets(j).Move After:=Sheets(j + 1)
End If
End If
Next j
Next i
End Sub
```

37. Trocar o nome de todas as abas

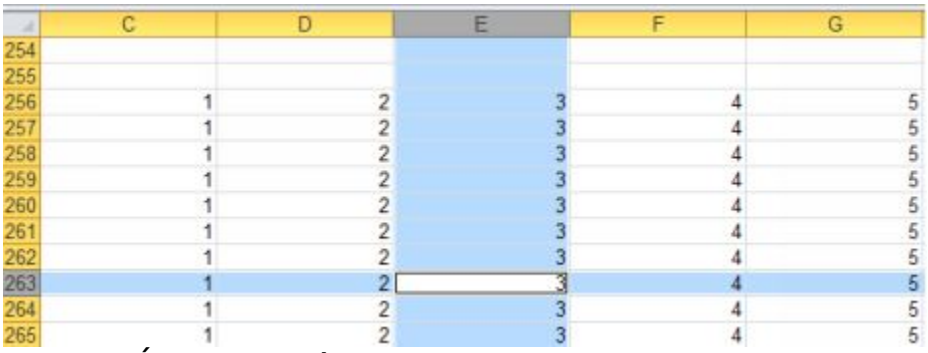
Esta macro tem o objetivo de alterar rapidamente o nome de todas as abas.

```
Sub TrocarNomeAba()
For Each Sheet In Worksheets
RotuloRetorno:
Sheet.Select
NovoNome = InputBox("Qual o novo nome para esta Aba ?")
```

```
If NovoNome = "" Then
    Exit Sub
End If
On Error GoTo RotuloRetorno:
ActiveSheet.Name = NovoNome
Next
End Sub
```

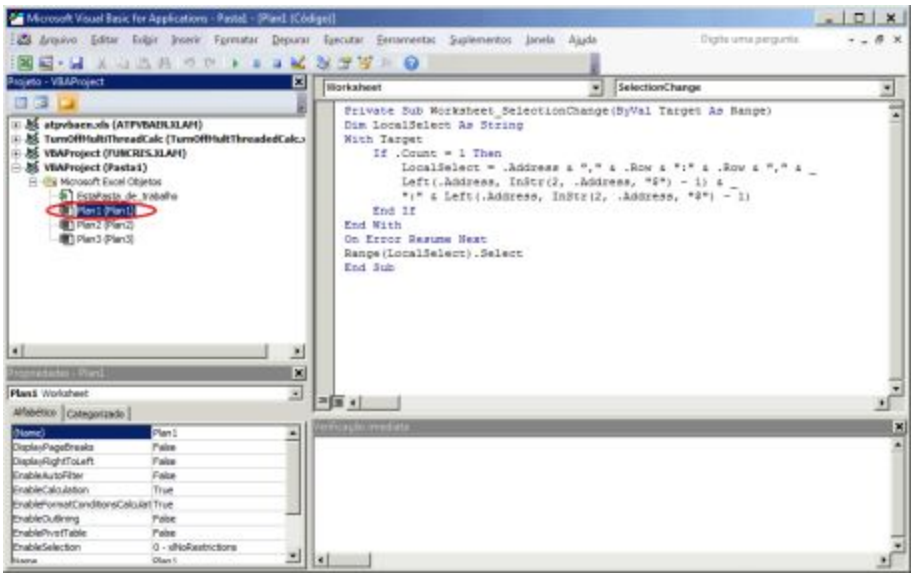
38. Destacar linha e coluna da célula selecionada

Esta macro é extremamente útil para usuários que precisam analisar grande quantidade de informações com frequência, a partir de uma célula selecionada, as colunas e linhas são destacadas, para facilitar a leitura de dados, conforme imagem abaixo:



	C	D	E	F	G
254					
255					
256	1	2	3	4	5
257	1	2	3	4	5
258	1	2	3	4	5
259	1	2	3	4	5
260	1	2	3	4	5
261	1	2	3	4	5
262	1	2	3	4	5
263	1	2	3	4	5
264	1	2	3	4	5
265	1	2	3	4	5

Obs1: É necessário incluir a macro dentro do objeto da planilha desejada, no exemplo abaixo a planilha que apresentará os destaques de linha e coluna será a “Plan1”.



```
Private Sub Worksheet_SelectionChange(ByVal Target As
Range)
Dim LocalSelect As String
With Target
    If .Count = 1 Then
        LocalSelect = .Address & "," & .row & ":" & .row & "," &
_
        Left(.Address, InStr(2, .Address, "$") - 1) & _
        ":" & Left(.Address, InStr(2, .Address, "$") - 1)
    End If
End With
On Error Resume Next
Range(LocalSelect).Select
End Sub
```

Interação com Windows

39. Salvar uma seleção como imagem

Esta macro, salva automaticamente uma seleção de células como uma imagem, o arquivo é salvo na mesma pasta com o mesmo nome da planilha ativa.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub SelectedRangeToImage()
    Dim iFilename As String
    Dim TempObjChart As Chart
    Dim Shp As Shape
    Dim Wsht As Worksheet
    Dim fileSaveName As Variant, pic As Variant
    Set Wsht = ActiveSheet
    Selection.Copy
    Wsht.Pictures.Paste.Select
    Set Shp = Wsht.Shapes(Wsht.Shapes.Count)
    Set TempObjChart = Charts.Add
    TempObjChart.ChartArea.Clear
    TempObjChart.Name = "PicChart" & (Rnd() * 10000)
    Set TempObjChart =
TempObjChart.Location(Where:=xlLocationAsObject,
Name:=Wsht.Name)
    TempObjChart.ChartArea.Width = Shp.Width
    TempObjChart.ChartArea.Height = Shp.Height
    TempObjChart.Parent.Border.LineStyle = 0
    Shp.Copy
    TempObjChart.ChartArea.Select
    TempObjChart.Paste
    iFilename = Application.ActiveWorkbook.Path & "\" &
ActiveSheet.Name & ".jpg"
    TempObjChart.Export Filename:=iFilename,
FilterName:="jpg"
    Wsht.Cells(1, 1).Activate
    Wsht.ChartObjects(Wsht.ChartObjects.Count).Delete
    Shp.Delete
```

End Sub

40. Converter todos os arquivos de uma pasta para PDF

Esta rotina automaticamente converte todos os arquivos do tipo Excel em uma pasta de origem, para PDF em uma pasta destino.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub PdfConvert()
Dim iNumArq As Integer
Dim iCounter As Integer
Dim sMyFiles() As String
Dim OriginFolder As String
Dim DestinyFolder As String
 OriginFolder = "C:\Pasta_Origem"
DestinyFolder = "C:\Pasta_Destino"
 FileFound = FindFiles(OriginFolder, sMyFiles, iNumArq, "*",
True)
If FileFound Then
    For iCounter = 1 To iNumArq
    Filename = sMyFiles(2, iCounter)
    ExtCount = Len(Filename) -
Application.WorksheetFunction.Search(".", Filename, 1) + 1
    Workbooks.Open (OriginFolder & "\" & Filename)
    Workbooks(Filename).Activate
    ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF,
Filename:= _
        DestinyFolder & "\" & Mid(Filename, 1, Len(Filename) -
ExtCount) & ".pdf", Quality:= _
        xlQualityStandard, IncludeDocProperties:=True,
IgnorePrintAreas:=False, _
```

```
        OpenAfterPublish:=False
    Workbooks(Filename).Close
    Next iCounter
End If
End Sub
Function FindFiles(ByVal sPath As String, ByRef sFoundFiles()
As String, _
    ByRef iArqEncontrados As Integer, _
    Optional ByVal sFileSpec As String = "*.*", _
    Optional ByVal blIncludeSubFolders As Boolean = False)
As Boolean
    Dim iCounter As Integer
    Dim sFileName As String
    Dim oFileSystem As Object, oParentFolder As Object,
oFolder As Object
    Set oFileSystem =
CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    Set oParentFolder = oFileSystem.GetFolder(sPath)
    If oParentFolder Is Nothing Then
        FindFiles = False
        On Error GoTo 0
        Set oParentFolder = Nothing
        Set oFileSystem = Nothing
        Exit Function
    End If
    sPath = IIf(Right(sPath, 1) = "\", sPath, sPath & "\")
    sFileName = Dir(sPath & sFileSpec, vbNormal)
    Do While sFileName <> ""
        iCounter = UBound(sFoundFiles, 2)
        iCounter = iCounter + 1
        ReDim Preserve sFoundFiles(1 To 2, 1 To iCounter)
        sFoundFiles(1, iCounter) = sPath
        sFoundFiles(2, iCounter) = sFileName
        sFileName = Dir()
    Loop
```

```
    If blIncludeSubFolders Then
        For Each oFolder In oParentFolder.SubFolders
            FindFiles oFolder.Path, sFoundFiles, iArqEncontrados,
sFileSpec, blIncludeSubFolders
        Next
    End If
    FindFiles = UBound(sFoundFiles, 2) > 0
    iArqEncontrados = UBound(sFoundFiles, 2)
    On Error GoTo 0
    Set oFolder = Nothing
    Set oParentFolder = Nothing
    Set oFileSystem = Nothing
End Function
```

1. Listar todos os arquivos de uma pasta

Esta macro é essencial para usuários que desejam listar todos os arquivos disponíveis em uma pasta dentro de um arquivo do Excel, seja para relatórios ou para verificações.

Obs1: Os arquivos serão listados na coluna A, iniciando pela primeira linha.
Obs2: Trocar a pasta origem em negrito.

```
Sub ListarArquivos()
Dim iNumArq As Integer
Dim iContador As Integer
Dim sMyFiles() As String
Dim pastaorigem As String
Dim pastadestino As String
pastaorigem = " C:\Users\UsuarioTeste\Documents "
ArqEncontrado = ProcurarArquivos(pastaorigem, sMyFiles,
iNumArq, "*", True)
If ArqEncontrado Then
    For iContador = 1 To iNumArq
    Filename = sMyFiles(2, iContador)
```

```
    ActiveSheet.Range("A" & iContador) = Filename
    Next iContador
End If
Columns(1).AutoFit
End Sub

Function ProcurarArquivos(ByVal sPath As String, ByRef
sFoundFiles() As String, _
    ByRef iArqEncontrados As Integer, _
    Optional ByVal sFileSpec As String = "*.*", _
    Optional ByVal blIncludeSubFolders As Boolean = False)
As Boolean
    Dim iContador As Integer
    Dim sFileName As String
    Dim oFileSystem As Object, oParentFolder As Object,
oFolder As Object
    Set oFileSystem =
CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    Set oParentFolder = oFileSystem.GetFolder(sPath)
    If oParentFolder Is Nothing Then
        ProcurarArquivos = False
        On Error GoTo 0
        Set oParentFolder = Nothing
        Set oFileSystem = Nothing
        Exit Function
    End If
    sPath = IIf(Right(sPath, 1) = "\", sPath, sPath & "\")
    sFileName = Dir(sPath & sFileSpec, vbNormal)
    Do While sFileName <> ""
        iContador = UBound(sFoundFiles, 2)
        iContador = iContador + 1
        ReDim Preserve sFoundFiles(1 To 2, 1 To iContador)
        sFoundFiles(1, iContador) = sPath
        sFoundFiles(2, iContador) = sFileName
        sFileName = Dir()
```

```
    Loop
    If blIncludeSubFolders Then
        For Each oFolder In oParentFolder.SubFolders
            ProcurarArquivos oFolder.Path, sFoundFiles,
iArqEncontrados, sFileSpec, blIncludeSubFolders
        Next
    End If
    ProcurarArquivos = UBound(sFoundFiles, 2) > 0
    iArqEncontrados = UBound(sFoundFiles, 2)
    On Error GoTo 0
    Set oFolder = Nothing
    Set oParentFolder = Nothing
    Set oFileSystem = Nothing
End Function
```

42. Copiar os arquivos de uma pasta para outra

Esta macro é muito utilizada para empresas e usuários que tenham acesso a uma rede, onde muitas vezes existe a rotina de realizar cópia de arquivos da rede para armazenar em uma outra pasta. Esta rotina é incrivelmente mais fácil com uma macro de Excel.

Obs: Substituir os valores de exemplo em negrito pelos valores desejados.

```
Sub CopiarColarArquivo()
Dim iNumArq As Integer
Dim iContador As Integer
Dim sMyFiles() As String
Dim pastaorigem As String
Dim pastadestino As String
pastaorigem = "
```
C:\Users\usuarioteste\Documents\Macro origem "
```
pastadestino = "
```
C:\Users\usuarioteste\Documents\Macro destino "

```
ArqEncontrado = ProcurarArquivos(pastaorigem, sMyFiles,
iNumArq, "*", True)
If ArqEncontrado Then
    For iContador = 1 To iNumArq
    FileCopy pastaorigem & "\" & sMyFiles(2, iContador),
pastadestino & "\" & sMyFiles(2, iContador)
    Next iContador
End If
End Sub
Function ProcurarArquivos(ByVal sPath As String, ByRef
sFoundFiles() As String, _
    ByRef iArqEncontrados As Integer, _
    Optional ByVal sFileSpec As String = "*.*", _
    Optional ByVal blIncludeSubFolders As Boolean = False)
As Boolean
    Dim iContador As Integer
    Dim sFileName As String
    Dim oFileSystem As Object, oParentFolder As Object,
oFolder As Object
    Set oFileSystem =
CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    Set oParentFolder = oFileSystem.GetFolder(sPath)
    If oParentFolder Is Nothing Then
        ProcurarArquivos = False
        On Error GoTo 0
        Set oParentFolder = Nothing
        Set oFileSystem = Nothing
        Exit Function
    End If
    sPath = IIf(Right(sPath, 1) = "\", sPath, sPath & "\")
    sFileName = Dir(sPath & sFileSpec, vbNormal)
    Do While sFileName <> ""
        iContador = UBound(sFoundFiles, 2)
        iContador = iContador + 1
        ReDim Preserve sFoundFiles(1 To 2, 1 To iContador)
```

```
        sFoundFiles(1, iContador) = sPath
        sFoundFiles(2, iContador) = sFileName
        sFileName = Dir()
    Loop
    If blIncludeSubFolders Then
        For Each oFolder In oParentFolder.SubFolders
            ProcurarArquivos oFolder.Path, sFoundFiles,
iArqEncontrados, sFileSpec, blIncludeSubFolders
        Next
    End If
    ProcurarArquivos = UBound(sFoundFiles, 2) > 0
    iArqEncontrados = UBound(sFoundFiles, 2)
    On Error GoTo 0
    Set oFolder = Nothing
    Set oParentFolder = Nothing
    Set oFileSystem = Nothing
End Function
```

Interações com Outolook

43. Enviar um e-mail simples com VBA

Esta macro tem por objetivo enviar um e-mail através de macro.

Obs 1: Alterar o assunto e a mensagem marcados em negrito
Obs 2: .CC e .BCC são opcionais, utilize apenas para os casos enviar cópias e cópias ocultas respectivamente.
Obs 3: O commando . **Send** marcado em negrito faz o e-mail ser enviado automaticamente, se trocar por .Display, o e-mail será apenas criado e deixado pronto, porém não será enviado.

```
Sub Send_Mail()
    Dim OutApp As Object
    Dim OutMail As Object
    Dim bMessage As String
    Set OutApp = CreateObject("Outlook.Application")
    Set OutMail = OutApp.CreateItem(0)
    bMessage = "Type the content line 1" & vbNewLine & _
            "Type the content line 2" & vbNewLine & _
            "Type the content line 3"
    On Error Resume Next
    With OutMail
        .to = " example@email.com "
        .CC = " copia@email.com "
        .BCC = " copiacega@email.com "
        .Subject = " Assunto do e-mail "
        .Body = " Texto do e-mail "
        . Send
    End With
    On Error GoTo 0
    Set OutMail = Nothing
    Set OutApp = Nothing
End Sub
```

44. Enviar arquivo como anexo por E-mail

Esta macro envia um e-mail incluindo o arquivo ativo como anexo deste e-mail.

Obs 1: Alterar o assunto e a mensagem marcados em negrito
Obs 2: .CC e .BCC são opcionais, utilize apenas para os casos enviar cópias e cópias ocultas respectivamente.
Obs 3: O commando . **Send** marcado em negrito faz o e-mail ser enviado automaticamente, se trocar por **.Display** ,

o e-mail será apenas criado e deixado pronto, porém não será enviado.

```
Sub SendWorkbookEmail()
    Dim OutApp As Object
    Dim OutMail As Object
    Set OutApp = CreateObject("Outlook.Application")
    Set OutMail = OutApp.CreateItem(0)
    On Error Resume Next
    With OutMail
        .to = " example@email.com "
        .CC = " copia@email.com "
        .BCC = " copiacega@email.com "
        .Subject = " Assunto do e-mail "
        .Body = " Texto do e-mail "
        .Attachments.Add (" C:\Arquivo_Exemplo.txt ")
        . Send
    End With
    On Error GoTo 0
    Set OutMail = Nothing
    Set OutApp = Nothing
End Sub
```

45. Enviar planilha ativa como anexo por E-mail

Esta macro envia um e-mail incluindo a planilha ativa como anexo deste e-mail.

Obs 1: Alterar o assunto e a mensagem marcados em negrito
Obs 2: .CC e .BCC são opcionais, utilize apenas para os casos enviar cópias e cópias ocultas respectivamente.
Obs 3: O commando . **Send** marcado em negrito faz o e-mail ser enviado automaticamente, se trocar por **.Display** ,

o e-mail será apenas criado e deixado pronto, porém não será enviado.

```
Sub SendActiveSheetEmail ()
    Dim Exten As String
    Dim FormtN As Long
    Dim OutApp As Object
    Dim OutMail As Object
    Dim OriginWKB As Workbook
    Dim DestWKB As Workbook
    Dim TempFilePath As String
    Dim TempFileFolder As String
    Application.ScreenUpdating = False
    Application.EnableEvents = False
    Set OriginWKB = ActiveWorkbook
    ActiveSheet.Copy
    Set DestWKB = ActiveWorkbook
    With DestWKB
        If Val(Application.Version) < 12 Then
            Exten = ".xls": FormtN = -4143
        Else
            Select Case OriginWKB.FileFormat
            Case 51: Exten = ".xlsx": FormtN = 51
            Case 52:
                If .HasVBProject Then
                    Exten = ".xlsm": FormtN = 52
                Else
                    Exten = ".xlsx": FormtN = 51
                End If
            Case 56: Exten = ".xls": FormtN = 56
            Case Else: Exten = ".xlsb": FormtN = 50
            End Select
        End If
    End With
    TempFilePath = Environ$("temp") & "\"
```

```
    TempFileFolder = "Part of " & OriginWKB.Name & " " &
Format(Now, "dd-mmm-yy h-mm-ss")
    Set OutApp = CreateObject("Outlook.Application")
    Set OutMail = OutApp.CreateItem(0)
    With DestWKB
        .SaveAs TempFilePath & TempFileFolder & Exten,
FileFormat:=FormtN
        On Error Resume Next
            With OutMail
                .to = " example@email.com "
                .CC = " copia@email.com "
                .BCC = " copiacega@email.com "
                .Subject = " Assunto do e-mail "
                .Body = " Texto do e-mail "
                .Attachments.Add (" C:\Arquivo_Exemplo.txt ")
                . Send
            End With
        On Error GoTo 0
        .Close savechanges:=False
    End With
    Kill TempFilePath & TempFileFolder & Exten
    Set OutMail = Nothing
    Set OutApp = Nothing
    Application.ScreenUpdating = True
    Application.EnableEvents = True
End Sub
```

46. Enviar e-mail com uma seleção como anexo

Esta rotina cria um arquivo a partir de uma seleção de células, este arquivo será enviado por e-mail através do Outolook.

Obs 1: Alterar o assunto e a mensagem marcados em negrito

Obs 2: .CC e .BCC são opcionais, utilize apenas para os casos enviar cópias e cópias ocultas respectivamente.
Obs 3: O commando . **Send** marcado em negrito faz o e-mail ser enviado automaticamente, se trocar por .Display, o e-mail será apenas criado e deixado pronto, porém não será enviado.

```
Sub Mail_Range()
    Dim wb As Workbook
    Dim iTempFolder As String
    Dim iTempFile As String
    Dim Source As Range
    Dim Dest As Workbook
    Dim iFormatNum As Long
    Dim iExt As String
    Dim OutApp As Object
    Dim OutMail As Object
    Set Source = Nothing
    On Error Resume Next
    Set Source = Selection.SpecialCells(xlCellTypeVisible)
    On Error GoTo 0
    If Source Is Nothing Then
        MsgBox "The source is out of range, please try again.",
vbOKOnly
        Exit Sub
    End If
    With Application
        .ScreenUpdating = False
        .EnableEvents = False
    End With
    Set wb = ActiveWorkbook
    Set Dest = Workbooks.Add(xlWBATWorksheet)
    Source.Copy
    Dest.Sheets(1).Cells(1).PasteSpecial Paste:=8
    Dest.Sheets(1).Cells(1).PasteSpecial Paste:=xlPasteValues
```

```
    Dest.Sheets(1).Cells(1).PasteSpecial
Paste:=xlPasteFormats
    Dest.Sheets(1).Cells(1).Select
    Application.CutCopyMode = False
    iTempFolder = Environ$("temp") & "\"
    iTempFile = "Selection of " & wb.Name & " " &
Format(Now, "dd-mmm-yy h-mm-ss")
    If Val(Application.Version) < 12 Then
        iExt = ".xls": iFormatNum = -4143
    Else
        iExt = ".xlsx": iFormatNum = 51
    End If
    Set OutApp = CreateObject("Outlook.Application")
    Set OutMail = OutApp.CreateItem(0)
    With Dest
        .SaveAs iTempFolder & iTempFile & iExt,
FileFormat:=iFormatNum
        On Error Resume Next
            With OutMail
                .to = " example@email.com "
                .CC = " copia@email.com "
                .BCC = " copiacega@email.com "
                .Subject = " Assunto do e-mail "
                .Body = " Texto do e-mail "
                .Attachments.Add (" C:\Arquivo_Exemplo.txt ")
                . Send
            End With
        On Error GoTo 0
        .Close savechanges:=False
    End With
    Kill iTempFolder & iTempFile & iExt
    Set OutMail = Nothing
    Set OutApp = Nothing
    With Application
        .ScreenUpdating = True
        .EnableEvents = True
```

```
    End With
End Sub
```

47. Enviar e-mail com arquivo externo como anexo

Esta macro envia automaticamente um e-mail com um arquivo anexado, é possível também combinar esta macro com as que foram apresentadas anteriormente, enviando desta maneira a planilha ou seleção juntamente com um outro arquivo anexado.

Obs 1: Alterar o assunto e a mensagem marcados em negrito
Obs 2: .CC e .BCC são opcionais, utilize apenas para os casos enviar cópias e cópias ocultas respectivamente.
Obs 3: O commando . **Send** marcado em negrito faz o e-mail ser enviado automaticamente, se trocar por .Display, o e-mail será apenas criado e deixado pronto, porém não será enviado.

```
Sub SendWorkbookEmail()
    Dim OutApp As Object
    Dim OutMail As Object
    Set OutApp = CreateObject("Outlook.Application")
    Set OutMail = OutApp.CreateItem(0)
    On Error Resume Next
    With OutMail
        .to = " example@email.com "
        .CC = " copia@email.com "
        .BCC = " copiacega@email.com "
        .Subject = " Assunto do e-mail "
        .Body = " Texto do e-mail "
        .Attachments.Add (" C:\Arquivo_Exemplo.txt ")
        . Send
    End With
```

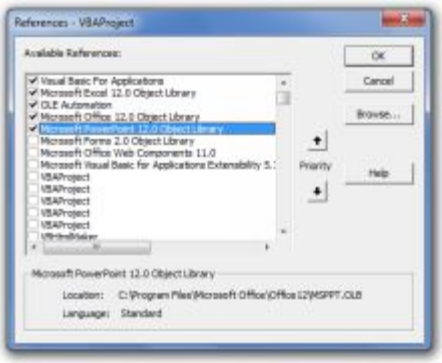
```
    On Error GoTo 0
    Set OutMail = Nothing
    Set OutApp = Nothing
End Sub
```

Interações com PowerPoint

48. Exportar gráficos para Microsoft PowerPoint

Esta macro automaticamente cria uma apresentação em PowerPoint com todos os gráficos dentro de uma planilha ativa.

Obs 1: É possível ajustar a ajustar a posição do gráfico no slide alterando o valor destaco em negrito
Obs 2: Dentro do editor VBA, clicar em ferramentas, referencias e então procurar por Microsoft PowerPoint Object Library, marque a opção na caixa de seleção e então pressione OK.




```
Sub Excel_chart_to_PPT ()
        Dim PptApp As PowerPoint.Application
        Dim iSlide As PowerPoint.Slide
        Dim ChartObj As Excel.ChartObject
        On Error Resume Next
```

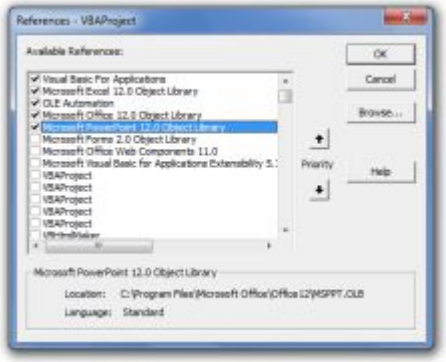
```
        Set PptApp = GetObject(, "PowerPoint.Application")
        On Error GoTo 0
        If PptApp Is Nothing Then
            Set PptApp = New PowerPoint.Application
        End If
        If PptApp.Presentations.Count = 0 Then
            PptApp.Presentations.Add
        End If
        PptApp.Visible = True
        For Each ChartObj In ActiveSheet.ChartObjects
            PptApp.ActivePresentation.Slides.Add
PptApp.ActivePresentation.Slides.Count + 1, ppLayoutText
            PptApp.ActiveWindow.View.GotoSlide
PptApp.ActivePresentation.Slides.Count
            Set iSlide =
PptApp.ActivePresentation.Slides(PptApp.ActivePresentation.
Slides.Count)
            ChartObj.Select
            ActiveChart.ChartArea.Copy
            On Error Resume Next

iSlide.Shapes.PasteSpecial(DataType:=ppPasteMetafilePictur
e).Select
            iSlide.Shapes(1).TextFrame.TextRange.Text =
ChartObj.Chart.ChartTitle.Text
            PptApp.ActiveWindow.Selection.ShapeRange.Left =
25
            PptApp.ActiveWindow.Selection.ShapeRange.Top =
150
            iSlide.Shapes(2).Width = 300
            iSlide.Shapes(2).Left = 600
        Next
    AppActivate ("Microsoft PowerPoint")
    Set iSlide = Nothing
    Set PptApp = Nothing
End Sub
```

49. Exportar seleção para Microsoft PowerPoint

Esta macro automaticamente exporta uma seleção para uma apresentação de PowerPoint

Obs 1: É possível ajustar a ajustar a posição da tabela que será exportada no slide alterando o valor destaco em negrito
Obs 2: Dentro do editor VBA, clicar em ferramentas, referencias e então procurar por Microsoft PowerPoint Object Library, marque a opção na caixa de seleção e então pressione OK.




```
Sub Selection_to_PowerPoint()
Dim iRange As Range
Dim PptObj As Object
Dim iPresent As Object
Dim iSlide As Object
Dim iShape As Object
  Set iRange = Selection
  On Error Resume Next
     Set PptObj = GetObject(class:="PowerPoint.Application")
     Err.Clear
     If PptObj Is Nothing Then Set PptObj =
CreateObject(class:="PowerPoint.Application")
     If Err.Number = 429 Then
```

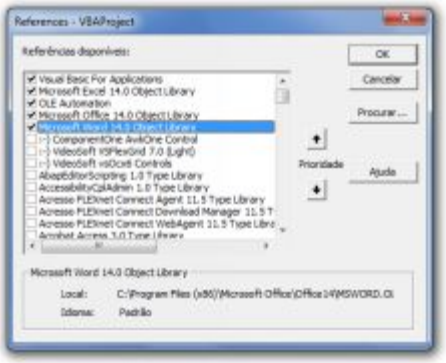
```
        MsgBox "PowerPoint could not be found, aborting."
        Exit Sub
      End If
  On Error GoTo 0
  Application.ScreenUpdating = False
  Set iPresent = PptObj.Presentations.Add
  Set iSlide = iPresent.Slides.Add(1, 11)
  iRange.Copy
  iSlide.Shapes.PasteSpecial DataType:=2
  Set iShape = iSlide.Shapes(iSlide.Shapes.Count)
     iShape.Left = 100
     iShape.Top = 160
  PptObj.Visible = True
  PptObj.Activate
  Application.CutCopyMode = False
  Application.ScreenUpdating = True
End Sub
```

Interações com Word

50. Exportar seleção para o Microsoft Word

Esta macro exporta automaticamente a seleção atual do Excel para o arquivo em Word aberto.

Obs: Dentro do editor VBA, clicar em ferramentas, referências e então procurar por Microsoft Word Object Library, marcar a caixa de seleção e pressionar OK.

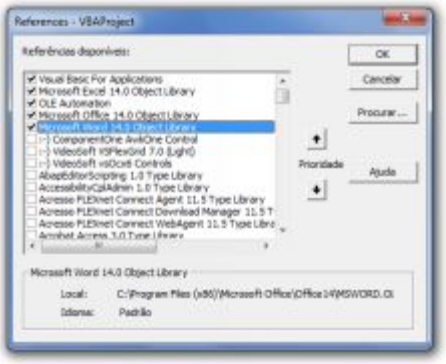



```
Sub CopyRangetoWord()
Dim WodAPP As Word.Application
Dim WordDOC As Word.Document
If Not TypeName(Selection) = "Range" Then
    MsgBox "Erro de seleção, tente novamente.",
vbExclamation, "Range Error"
Else
    Set WodAPP = GetObject(, "Word.Application")
    Set WordDOC = WodAPP.ActiveDocument
    Selection.Copy
    WodAPP.Selection.PasteSpecial Link:=False,
DataType:=wdPasteRTF, _
        Placement:=wdInLine, DisplayAsIcon:=False
    Set WordDOC = Nothing
    Set WodAPP = Nothing
End If
End Sub
```

1. Exportar dados para o Microsoft Word

Esta rotina exporta as informações de uma seleção para um novo arquivo do Word, salva o arquivo com o mesmo nome do Excel e também no mesmo diretório.

Obs: Dentro do editor VBA, clicar em ferramentas, referências e então procurar por Microsoft Word Object Library, marcar a caixa de seleção e pressionar OK.




```
Sub Worksheet_to_Word()
    Dim Question As Integer
    Question = MsgBox("Esta macro vai exportar toda
seleção para word, deseja continuar ?", vbYesNo)
    If question <> 6 Then
    Exit Sub
    End If
    Set Object = CreateObject("Word.Application")
    Object.Visible = True
    Set newObject = Object.Documents.Add
    ActiveSheet.UsedRange.Copy
    newObject.Range.Paste
    Application.CutCopyMode = False
    Object.Activate
    On Error Resume Next
    newObject.SaveAs
Filename:=Application.ActiveWorkbook.Path & "\" &
ActiveSheet.Name
End Sub
```