VISUAL BASIC FOR APPLICATIONS

WORD 2010 VBA

Crie macros e aplicações com este poderoso recurso do Word.



APRESENTAÇÃO

Atualmente exige-se de muitos profissionais que trabalham com softwares do pacote Office, como Excel, Word, Access e PowerPoint, o conhecimento de programação em VBA (Visual Basic for Applications) devido ao fato dessa poderosa linguagem ajudar a solucionar problemas complexos que, muitas vezes, não poderiam ser resolvidos sem o uso da programação.

Encontramos na Internet muitos livros, apostilas, tutoriais e dicas sobre Excel VBA e Access VBA, mas quando se refere ao VBA do Word o material é bastante escasso, por esse motivo resolvi criar este e-book para mostrar que também no Word, através do VBA, podemos criar inúmeras soluções e automatizar várias tarefas, portanto, este e-book tem como objetivo mostrar ou até mesmo ensinar aos usuários do Microsoft Word a utilizar o VBA para adicionar poderosos recursos ao Word.

Nesta obra estarei utilizando uma linguagem simples, sem muitos termos técnicos, pois ela é voltada para usuários leigos, de nível iniciante e/ou intermediário no que se refere a conhecimento em VBA e está organizada em cinco capítulos, sendo que o primeiro é uma breve introdução ao VBA, onde o leitor estará se familiarizando com o ambiente de desenvolvimento (Visual Basic for Applications) que vem embutido no Word, já no segundo capítulo apresentarei também de forma sucinta como criar uma interface amigável para interagir com o usuário, e no terceiro capítulo estarei mostrando, através de alguns exemplos, como criar macros através do VBA, e no quarto, apresentarei também alguns projetos e soluções, onde o leitor poderá acompanhar o desenvolvimento passo a passo de cada um destes projetos, mostrando na prática como usar os principais recursos de programação do Word/VBA, e por fim no último capítulo são disponibilizado três tabelas (Ascii, Operadores matemáticos e Cores RGB).

Todas as Macros e Aplicações aqui apresentadas foram criadas utilizando o Microsoft Word na sua versão 2010, uma vez que esta é a versão mais utilizada atualmente, contudo, também foram testadas na versão 2007 e não apresentaram nenhum problema.

Em se tratando de programação não existe “uma maneira correta de fazer isso”, o ideal é procurar deixar o código mais enxuto possível e fazer com que tudo que funcione de forma prática, portanto, peço ao leitor que encare os projetos aqui apresentados como ponto de partida e/ou sugestões para que através deles o leitor possa ter ideias e criar novos projetos ou adaptá-los às suas necessidades.

Acompanhe e aprenda a desenvolver com o Word/VBA.

AUTOR

Ednilson Muniz Mendes,
Técnico de Informática: Possui mais de 15 anos de conhecimento e experiência em MS Office, automatização de Documentos, Aplicações e Soluções em VBA (Visual Basic For Applications).

1. - INTRODUÇÃO AO VBA 7

1.1-ACESSANDO O EDITOR DO VBA 8
1.2-APRESENTANDO O EDITOR DO VBA 10
1.3-AS CORES DO VBA 14
1.4-GERANDO CÓDIGOS 15
1.5-DELARAÇÃO DE VARIÁVEL 16
Regras para nomear as variáveis em VBA 16
Tipos de Variáveis 16
Declarando uma variável. 17
Variável do Tipo Objeto 17
Declaração da Variável Objeto 17
Atribuição de uma variável Objeto 17
Variáveis - Declaração Opcional e o tipo Variant 18
Os Prós e Contras da Utilização do tipo Variants 18
Variáveis – Declaração Forçada 18
1.6-TRATAMENTOS DE ERROS 19
Tipos de Erros 19
Respostas aos erros 19

2-INTERFACE 22

2.1-A Função MsgBox 22
2.2-A Função InputBox 24
2.3-Criando Formulários 25

3.-O QUE É MACRO? 28

3.1-Criando uma saudação ao usuário 29
3.2-Exibindo uma saudação ao usuário de acordo com o horário 34
3.3-Saudação – interagindo com usuário 40
3.4-Outro exemplo utilizando a função INPUTBOX 45
1ª Opção: usando Controle ActiveX 48
2ª Opção: Adicionando na Faixa de Opções 52
3ª Opção: Adicionando à Barra de Ferramentas de Acesso Rápido. 59
3.5-Inserindo o conteúdo de uma InputBox no documento. 65
3.6-Exibindo uma mensagem na barra de status 73
3.7-Visualizar Impressão. 77
3.8-Contando caracteres de um texto ou palavra selecionada. 82
3.9-Colocando borda em todas as imagens do documento. 88
3.10-Colocando borda nas imagens selecionadas. 94
3.11-Usando a Função StrReverse 101
1ª Opção: Exibindo o texto em uma caixa de mensagem 101
2º Opção: Exibindo o texto no documento 104
3.12-Extraindo E-mails de um documento. 109
3.13-Exibir em Tela cheia sem a barra de ferramentas. 116
3.14-Criando pastas através do VBA 123
3.15-Remover Links de um documento. 129

4.- PROJETOS 138

4.1-PROJETO - 01 139
Criador de Mini Banners 139
4.2-PROJETO 02 219
Procura Texto 219
4.3-PROJETO 03 238
Prazo Exato 238
3.4-PROJETO 04 263
Calculadora 263
4.5-PROJETO 05 286
Declaração de Matrícula 287
4.6-PROJETO 06 316

Cálculos Diversos ... 316
4.7-PROJETO 07 ... *380*
Nota de Orçamento ... 380
4.8-PROJETO 08 ... *441*
Estados Brasileiros e suas Capitais ... 441
4.9-PROJETO 09 ... *509*
Navegador ... 510
4.10-PROJETO 10 ... *536*
Capa Para Trabalhos Acadêmicos ... 536

6-TABELA ASCII COMPLETA ... 572

7-OPERADORES MATEMÁTICOS E LÓGICOS DO VBA ... 580

8-TABELA DE CORES RGB ... 581

9-DOWNLOAD DOS ARQUIVOS. ... 596

CAPÍTULO - I

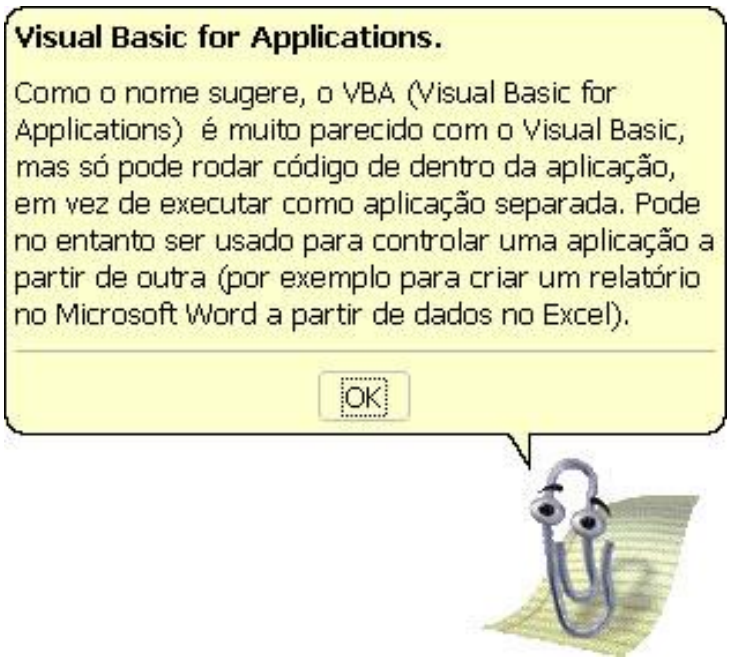
CONHECENDO O VBA

1. - INTRODUÇÃO AO VBA

Até a versão 7.0 o Word utilizava o WordBasic, mas a partir da versão 8.0 (Word 97) o WordBasic foi substituído pelo VBA (Visual Basic for Applications).

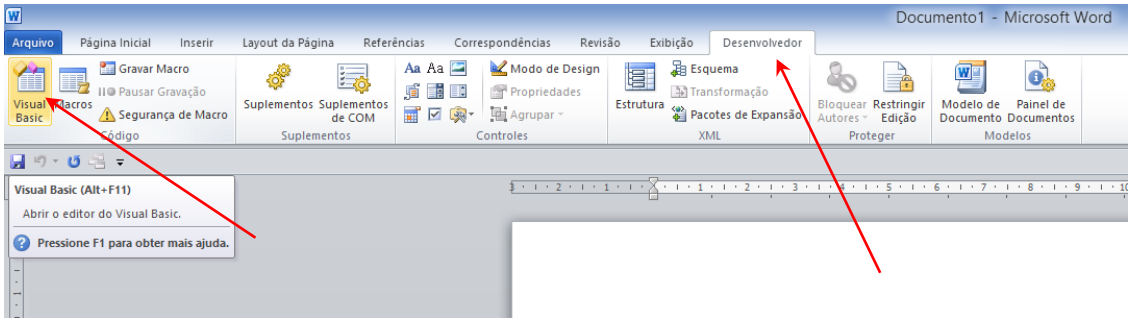
O Visual Basic for Applications é uma linguagem de programação interna, ligada a todos os aplicativos do pacote Office, como o Access, Word, Excel, PowerPoint e outros. O Visual Basic for Applications, também chamado de VBA, propicia o desenvolvimento de ferramentas e rotinas que proporcionam ao Office recursos quase ilimitados.

Como já foi dito, o Visual Basic For Applications é um ambiente de programação que vem embutido em diversos aplicativos e possui grande parcela dos recursos do Visual Basic (Pacote completo da linguagem vendido como produto independente), porém com uma limitação fundamental, ele não gera aplicativos autônomos (executáveis). Mas embora o VBA não gere nenhum executável, podemos com ele customizar tarefas, produzir soluções complexas e criar verdadeiras aplicações.

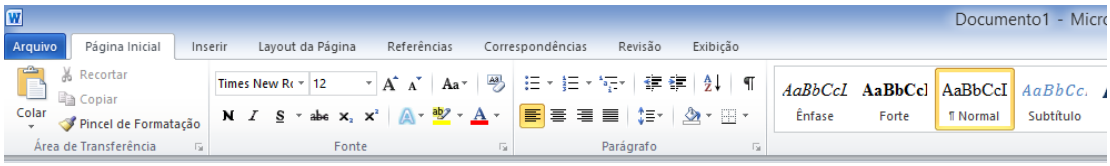


1.1-ACESSANDO O EDITOR DO VBA

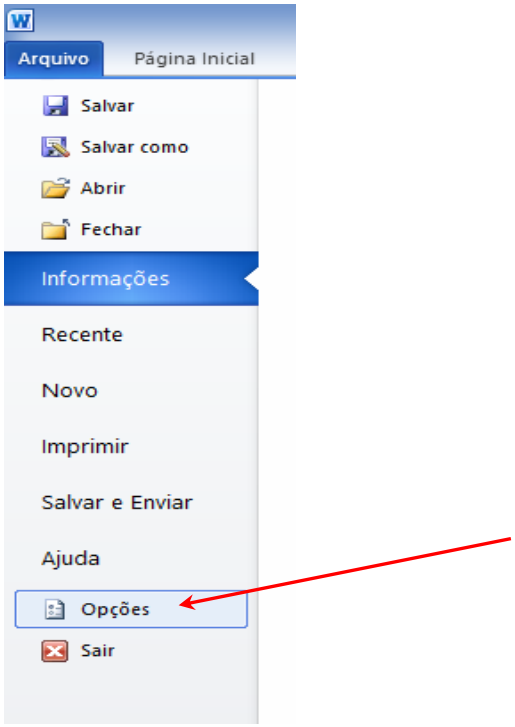
Para exibir o ambiente de desenvolvimento do VBA clique na guia **Desenvolvedor** e no bloco **Código** clique em **Visual Basic** ou se preferir use as teclas de atalho **ALT + F11**.



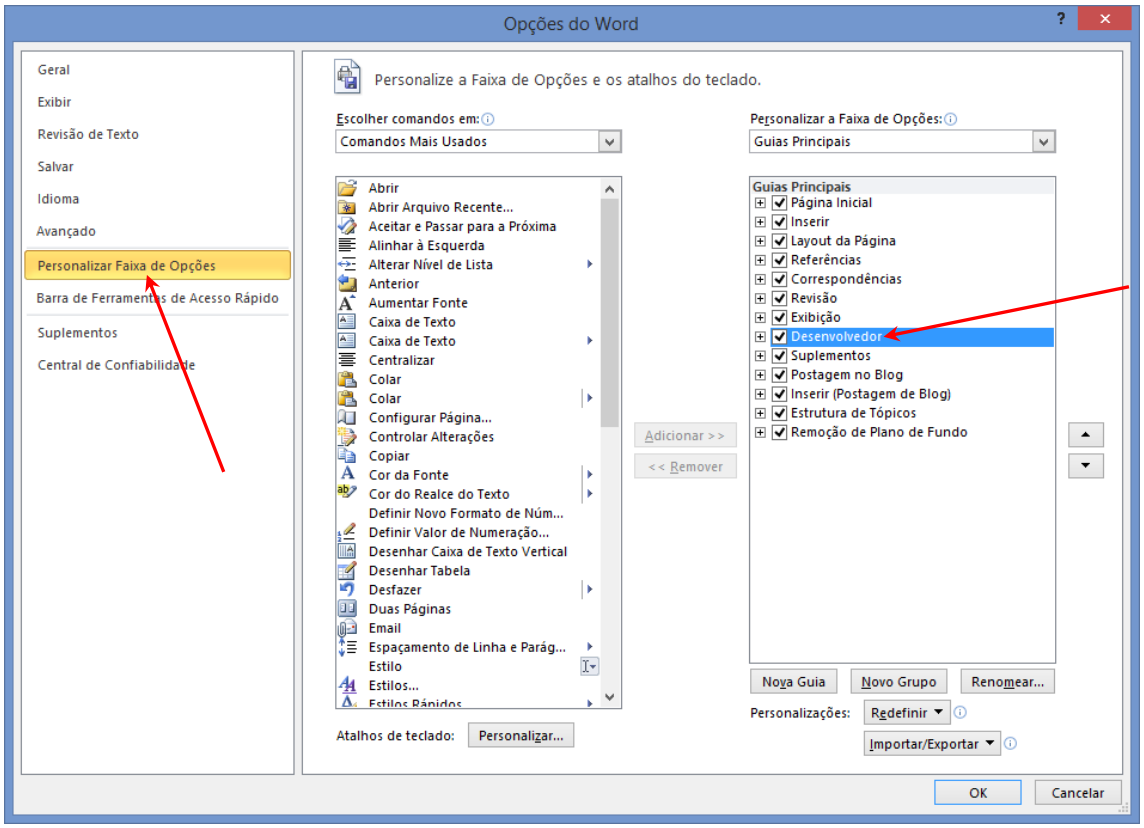
Se a Guia **Desenvolvedor** não estiver visível



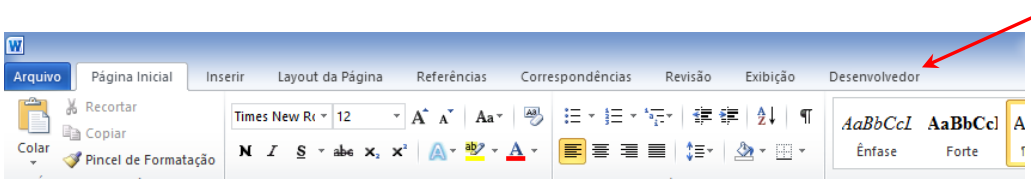
Acione a Guia **Arquivo** e clique em **Opções...**



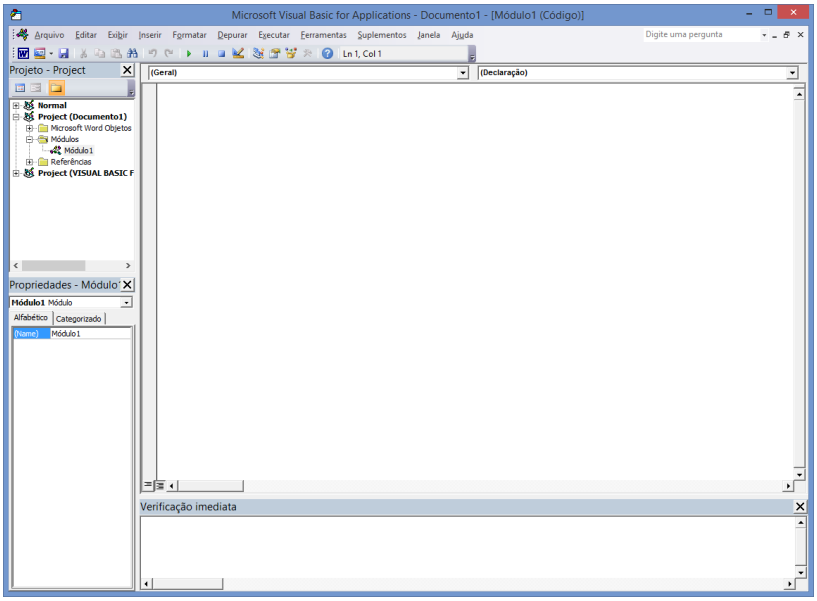
Em seguida clique em **Personalizar Faixa de opções**... ...e em Guias Principais selecione **Desenvolvedor**.



Feito isso a guia **Desenvolvedor** será exibido, como mostra a imagem abaixo:

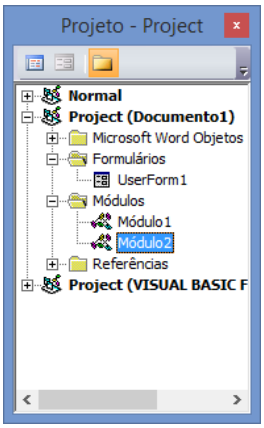


1.2-APRESENTANDO O EDITOR DO VBA

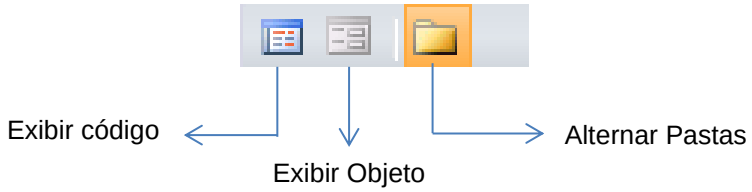


O ambiente de desenvovimento do VBA, além de uma barra de menu e de uma barra de ferramentas, possui algumas janelas, cuja as principais são:

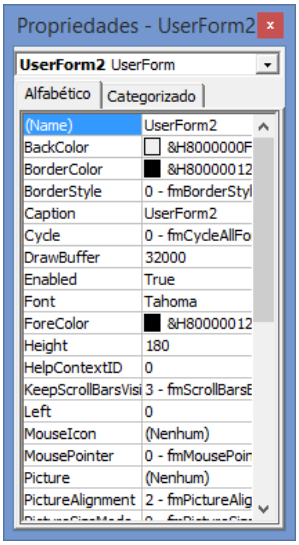
Janela de Proejto ou Project Explorer: Exibe todos os elementos que fazem parte de um ou mais proejtos (Ex: Documentos, banco de Dados, Formulários, módulos e módulas de classe). Com essa janela podemos ver a hierarquia existente em um projeto VBA que está em edição no momento.



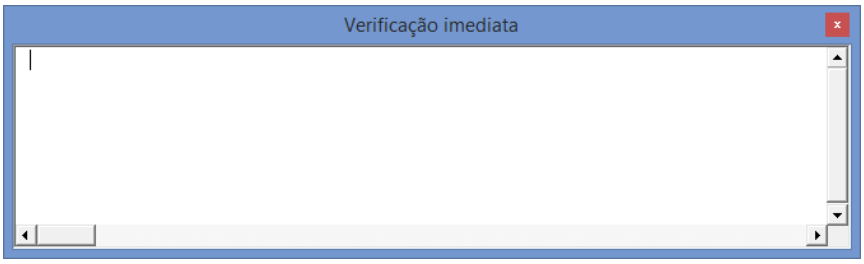
Essa janela também possui três ícones que alteram o modo de visualização do objeto selecionado.



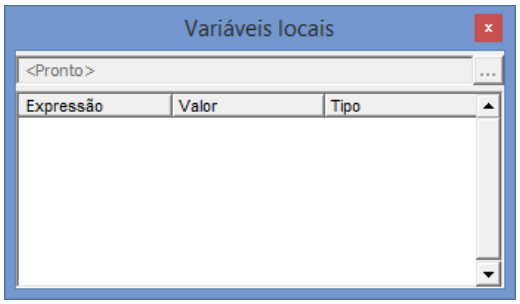
Janela Propriedades: Exibe as definições de propriedades de um objeto selecionado num formulário (Ex: o próprio formulário, caixas de textos, botões de comando, caixa de listagem, etc). Você pode alterar estas propriedades em tempo de criação. Quando você seleciona múltiplos controles, a janela Propriedades contém uma lista das propriedades comuns a todos os controles selecionados.



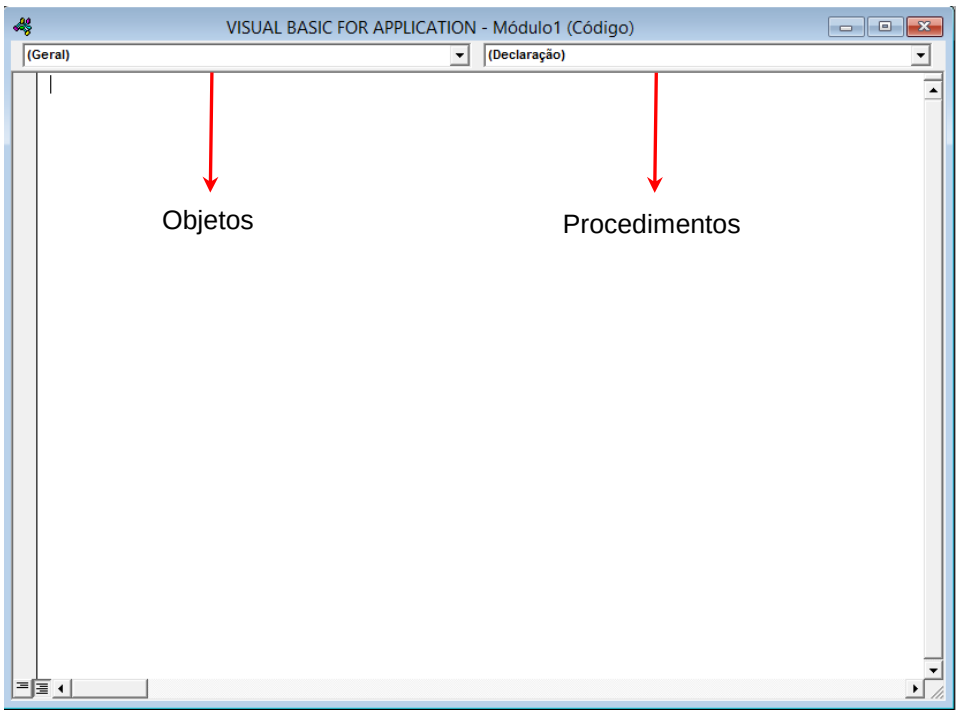
Janela Verificação Imediata: Verifica o valor de uma variável, executa programas no ambiente VBA ou fora dele.



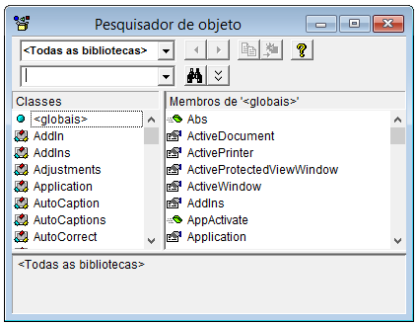
Janela de Variáveis Locais: Exibe automaticamente todas as variáveis declaradas e seus valores no procedimento atual.



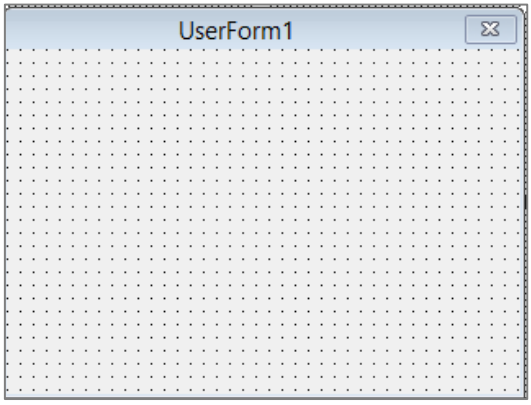
Janela de Código: Exibe e edita código de Visual Basic. Você pode abrir tantas janelas de código quantas forem os módulos, de modo que possa visualizar facilmente o código em diferentes formulários ou módulos e copiar e colar entre eles. Podemos dizer que módulo é a unidade básica de um código em VBA e todo módulo possui procedimentos, que são as normas para a condução da tarefa.



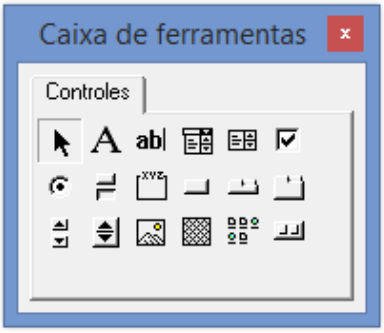
Janela Pesquisador de Objeto: Exibe as classes, propriedades, métodos, eventos e constantes disponíveis das bibliotecas de objetos do Word e do VBA e os procedimentos no seu projeto. Você pode usá-lo para localizar e usar objetos criados por você e objetos de outros aplicativos.



Inserir Userform : Permite-lhe criar as interfaces ou caixas de diálogo no seu projeto. Você desenha e visualiza controles em um formulário. Quando esta janela está ativa a caixa de ferramentas é automaticamente exibida. Utilize os botões da Caixa de ferramentas para inserir controles no formulário.



Caixa de Ferramentas: Exibe os controles com os quais se desenham os formulários, ou seja, nesta caixa estão os controles que podem ser adicionados aos formulários para criar a interface dos aplicativos.



1.3-AS CORES DO VBA

Na Janela de Edição são programados, armazenados e exibidos os procedimentos VBA criados.

Porém, o texto envolvido em cada procedimento é dotado de uma série de cores diferentes, sendo que cada uma tem significado diferente.

Cor	Significado
Azul	Palavras-Chave da linguagem. Ex: Sub, End Sub, Function, If, Then, Else, While, Loop,…
Vermelho	Sempre que escreve na janela uma linha em linguagem VBA, o editor vai verificar a sintaxe da linguagem por forma a indicar se existe ou não algum erro de escrita. Se for detectado um erro a linha fica a vermelho e é exibida uma mensagem de erro, senão todas as palavras são reconhecidas, coloridas da cor da respectiva categoria e alteradas para letras maiúsculas ou minúsculas, conforme esteja pré-definido.
Preto	Nome de variáveis, procedimentos, valores, operadores,…
Verde	Comentários introduzidos no seio dos procedimentos. Estes comentários servem para o utilizador poder associar algumas explicações aos procedimentos realizados. De referir que as palavras com esta cor são ignoradas no procedimento, i.e., não produzem qualquer efeito na sua execução. Para introduzir comentários bastará que o caractere ‘ anteceda o texto a introduzir.
Amarelo	Um sombreado amarelo poderá aparecer sobre a linha que identifica um procedimento. Esta cor simboliza a ocorrência de um erro na execução do respectivo procedimento e o estado de execução do mesmo, i.e., o procedimento iniciou a execução, durante a qual detectou um erro e agora está parado, mas ainda em execução. Quando isto acontecer não deverá voltar a dar ordem de execução do procedimento, sem antes parar (Stop) a execução mal sucedida.

1.4-GERANDO CÓDIGOS

Antes de começarmos a gerar os primeiros códigos é necessário termos bem definidos alguns conceitos utilizados em linguagem de programação, como: propriedade, instruções, métodos, objetos, declaração de variáveis, tratamento de erros e outros.

Propriedades: definem tanto a aparência (tamanho, cor, etc) como também a posição de um objeto.

Instruções ou Procedimentos: é um conjunto de códigos dispostos de forma lógica dentro do Editor do VBA, com a finalidade de executar determinada tarefa. No VBA todos os procedimentos devem começar com a instrução SUB ou FUNCTION e terminar com END SUB ou END FUNCTION.
Veja um exemplo:

```
Sub exemplo_de_mensagem()
 MsgBox "Olá seja bem vindo", vbOKOnly, "Mensagem ao Usuário"
End Sub
```

Objetos: é qualquer controle que pode ser manipulado, ou seja, pode ser controlado por meio da linguagem VBA e que é utilizado na execução de tarefas. Exemplo: barras de ferramentas, caixas de texto, caixas de diálogo etc.

Declaração de variáveis: uma variável é um valor que fica armazenado na memória do computador enquanto o procedimento está sendo executado, sendo removida da memória assim que o procedimento termina.

Porque devemos declarar uma variável?
Embora a declaração de uma variável não seja obrigatória no VBA, ela é altamente recomendável para agilizar o sistema informando que determinada informação será armazenada. É importante declarar uma variável principalmente quando se necessita de utilizar um valor especifico inúmeras vezes e também pelo fato de que a execução do código será mais rápido.

1.5-DELARAÇÃO DE VARIÁVEL

Regras para nomear as variáveis em VBA

Ao nomear uma variável devem-se obedecer sempre as seguintes regras:

- Não utilize pontos, espaços, vírgulas, pontos de exclamação ou símbolos especiais (#, @, $, &, %, etc);
- O nome deve ter menos de 255 caracteres;
- Não utilize palavras reservadas do sistema, por exemplo no Word, não podemos empregar termos como Document, ThisDocument, Formfields, etc;
- Todo nome de variável deve iniciar com uma letra.

Tipos de Variáveis

O tipo de variável está associado ao gênero de informação que esta quer armazenar.

Boolean – 2 bytes – Permite armazenar valores Boolenaos – True ou False

Byte – 1 Byte – permite armazenar números sem sinal entre 0 e 255

Currency - 8 bytes – permite armazenar moeda

Date – 8 Bytes – permite armazenar datas

Double – 8 bytes – permite armazenar um real desde -1.79769313486232E308 até -4.94065645841247E-324 para valores negativos, e desde 1.79769313486232E308 até 4.94065645841247E-324 para valores positivos.

Single *– 4 bytes – permite armazenar um real desde -3.402823E38 até -1.4011298E-45, para valores negativos e desde 3.402823E38 até 1.4011298E-45, para valores positivos*

Integer – 2 bytes - permite armazenar números inteiros entre -32.768 e 32767

Long – 4 bytes – permite armazenar números inteiros entre -2 147 483 648 e 2 147 483 648

Object – 4 bytes – utilizado para fazer referência a um objeto do Word

String – 1 byte por caractere – permite armazenar conjuntos de caracteres

Variant – 16 bytes - permite armazenar qualquer tipo de dados

User-Defined – permite armazenar valores de tipos diferentes

Declarando uma variável.

Dim numer **As Integer**

Onde:

Dim – Palavra chave que indica uma declaração de variáveis (abreviatura de dimensão)
numer - nome da variável a utilizar
As – palavra chave utilizada para separar o nome da variável do tipo de dados
Integer – tipo de dados atribuído à variável

É possível na mesma declaração de variáveis declarar variáveis de diversos tipos:

Dim numer_1 **As Integer**, numer_2 **As Date**, numer_3 A**s Double**

Para declarar diversas variáveis do mesmo tipo:

Dim numer_1, numer_2, numer_3 **As Currency**

Variável do Tipo Objeto

Uma variável objeto representa uma referência a um objeto. Uma variável de extrema importância que facilita a codificação e melhora a performance da subrotina.

Declaração da Variável Objeto

Dim obj **As Object**

Atribuição de uma variável Objeto

Set obj = <Objecto>

Onde:

Set – palavra chave que indica a assignação de uma variável objecto
obj - Variável Objecto
= - Operador de assignação
<Objecto> - Objeto a ser atribuído à variável

Variáveis - Declaração Opcional e o tipo Variant

A declaração de variáveis é opcional, se as variáveis não forem declaradas o VBA faz a sua declaração por defeito. Assim sempre que a instrução do Dim é omitida para uma variável, essa assume o tipo Variant.

Os Prós e Contras da Utilização do tipo Variants

Prós

- Diminui o número de linhas de código
- Não é necessário estar preocupado se a variável está ou não declarada porque o VBA automaticamente o faz

Contras

- Aumenta o tempo de execução – o VBA primeiro precisa reconhecer o tipo de dados com os quais está a trabalhar.
- Este tipo de dados consome mais memória (uma vez que tem que alocar espaço para qualquer tipo de dados que lhe seja atribuído) – 16 bytes mais um byte por caractere se for String => problemas de performance para subrotinas grandes.
- Não é possível saber o tipo de dados que uma determinada variável contém – dificultando a detecção de erros.

Variáveis – Declaração Forçada

Para que o VBA detecte um erro sempre que uma variável não seja declarada deverá fazer:

Menu Ferramentas >>Opções... >> na guia **Editor** selecione **Requerer declaração de variável;**

Ou então, escrever no início de cada módulo **Option Explicit**

Neste caso sempre que for detectada uma variável que ainda não foi declarada dará uma mensagem de erro - **Variable Not Defined**

1.6-TRATAMENTOS DE ERROS

Embora seja tomado todo cuidado para evitar erros durante a execução dos códigos, acabamos sempre nos deparando com alguns erros.
O VBA, assim como outras linguagens de programação, possuem recursos que permitem identificar os erros e controlar o fluxo de execução do código.

Tipos de Erros

No tempo da compilação: caracterizados por falhas na sintaxe (falta de aspas, parênteses, pontos, vírgulas, etc).

No tempo da execução: ocasionados pelas tentativas de executar uma operação inválida, por exemplo, uma referência a um arquivo excluído ou inexistente.

Erros lógicos: quando resultados incorretos são produzidos pelo aplicativo.

Respostas aos erros

Todo procedimento que provoca um erro inclui uma instrução **On Error**. A instrução On Error ativa uma rotina de tratamento de erros e especifica o local da rotina dentro de um procedimento; também pode ser usado para desativar uma rotina de tratamento de erros.

Em uma rotina, quando o erro ocorrido é identificado, uma mensagem é exibida ao usuário.

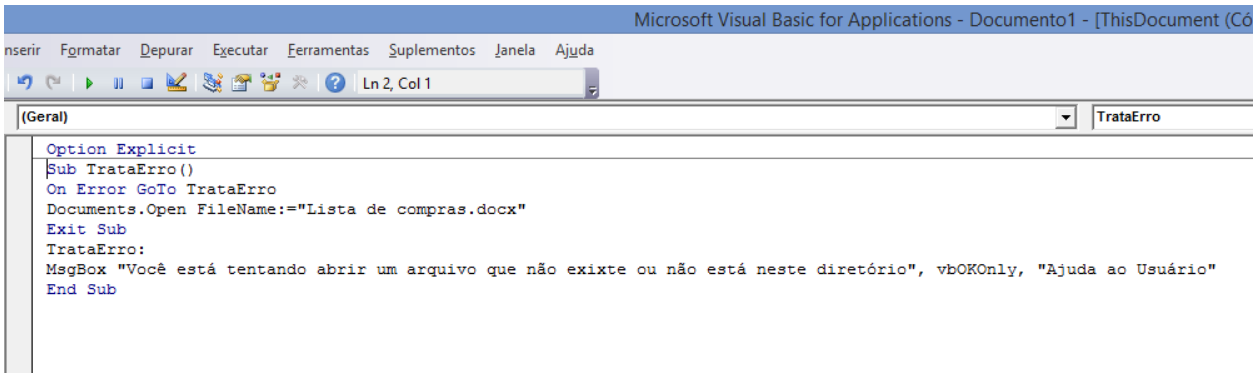
Instrução	*Definição*
On Error GoTo Linha	Em caso de erro, vá para a linha que ativará a rotina de gerenciamento de erros.
On Error Resume Next	Especifica que, quando ocorre um erro de tempo de execução, controle vai para a declaração imediatamente seguinte a instrução em que ocorreu o erro e continua de execução a partir desse ponto.
On Error GoTo 0	Desativa o tratamento de erros ativado no procedimento atual.

Para sair, é preciso dar os comandos **Resume** ou **Exit**, sendo que o primeiro retorna o controle do programa para o procedimento principal e o segundo sai do procedimento.

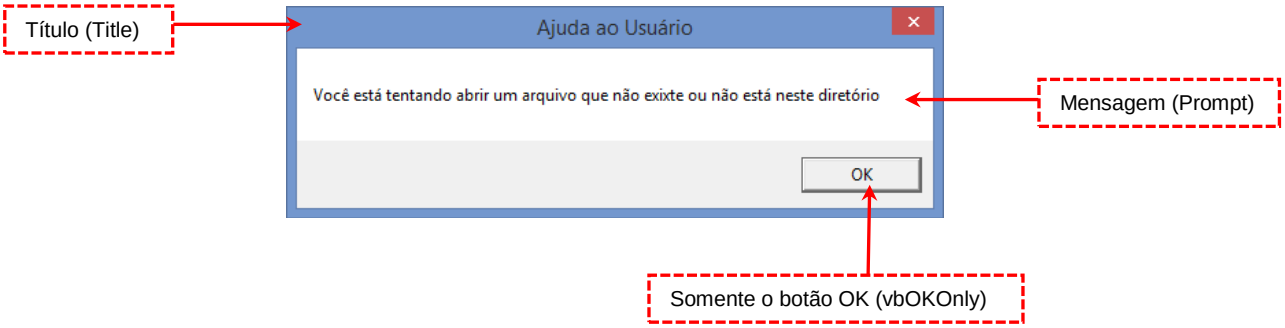
Eis um exemplo de uma rotina de tratamento de erro.

```
Option Explicit
Sub TrataErro( )
  On Error GoTo  TrataErro
     Documents.Open FileName:=”Lista de compras.docx”
  Exit Sub
  MsgBox “Você está tentando abrir um arquivo que não existe ou não está neste diretório”, vbOKOnly, “Ajuda ao Usuário”
End Sub
```

Veja como fica essa rotina no editor de código.



Essa é a mensagem que será exibida quando ocorrer um erro.



CAPÍTULO - II

INTERFACE COM O USUÁRIO

2-INTERFACE

A Interface com o Usuário é fundamental em qualquer aplicação, pois, é a parte visível pelo qual ele se comunica com o usuário para realização das suas tarefas.

O VBA possui modelos prontos de caixas de mensagens, com as quais podemos usá-las para guiar e ajudar o usuário na solução de problemas.

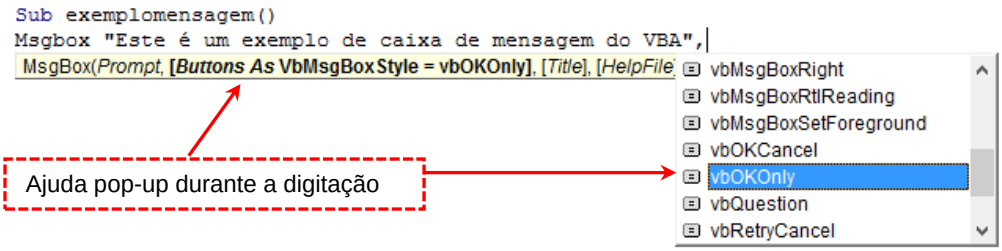
2.1-A Função MsgBox

Essa função é muito utilizada para exibir uma mensagem ao usuário, sendo que sua sintaxe é:

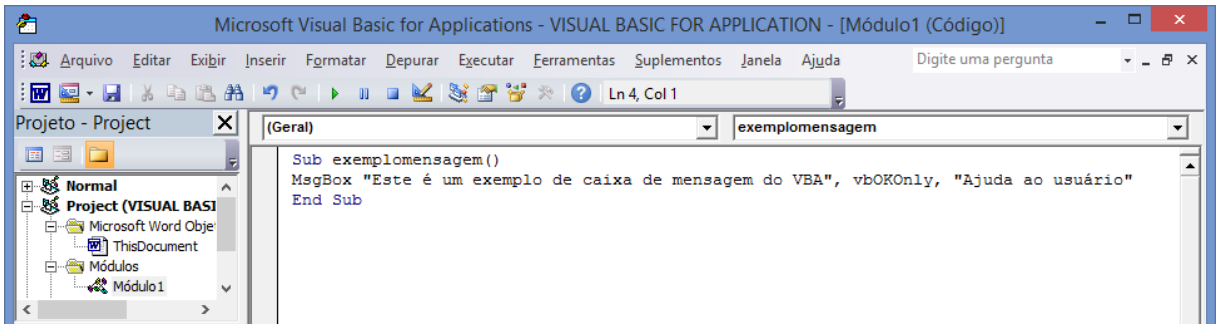
MsgBox “mensagem ao usuário”, tipo de caixa e/ou tipo de botão, “Titulo da caixa”.

O legal do ambiente de programação do VBA é sua ajuda pop-up que surge no momento que estamos escrevendo os códigos.

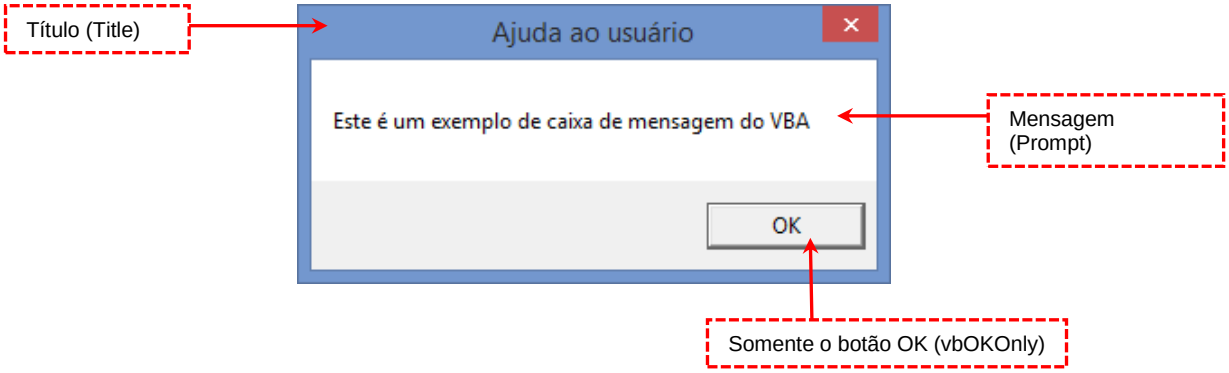
Observe a imagem abaixo:



Veja um exemplo de uma caixa de mensagem usando a função MsgBox, para isso criaremos uma macro chamada **exemplomensagem**.



Quando o código acima é executado exibe a seguinte mensagem:

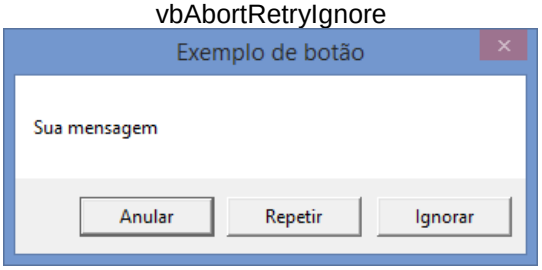


Para facilitar a comunicação com o usuário, o VBA possui botões prontos para sua caixa de mensagem e cada um desses botões tem um número correspondente, bem como um código VBA. Veja na tabela abaixo:

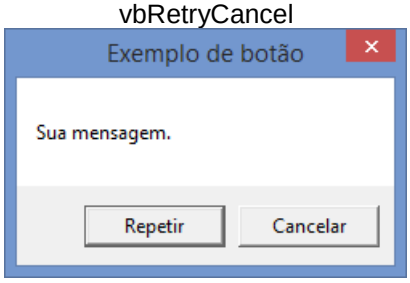
Botão	*Valor*	*Código VBA*
OK	1	vbOK
Cancelar	2	vbCancel
Anular	3	vbAbort
Repetir	4	vbRetry
Ignorar	5	vbIgnore
Sim	6	vbYes
Não	7	vbNo

Veja alguns exemplos:

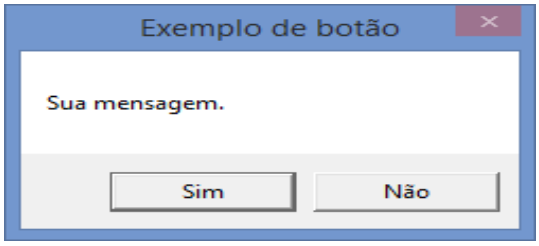
vbAbortRetryIgnore



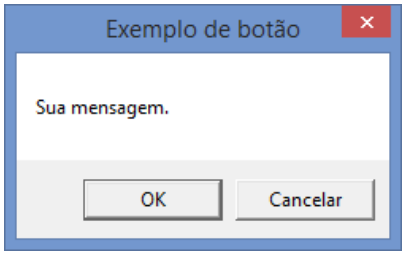
vbRetryCancel



vbYesNo



vbOKCancel



2.2-A Função InputBox

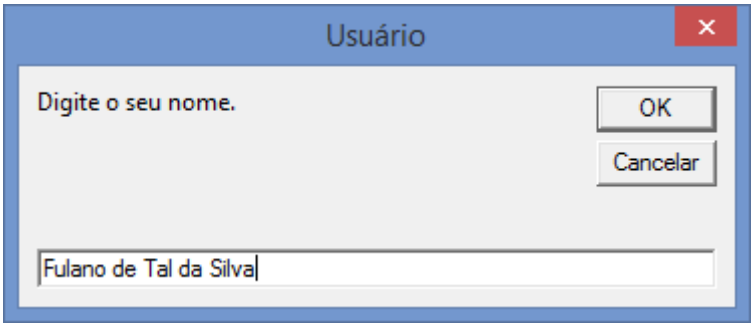
É uma caixa de mensagem mais interativa, pois nela o usuário pode inserir dados. Sua sintaxe é:

InputBox (“mensagem”, “Titulo”, “Resposta Padrão”, “Posição X”, “Posição Y”)

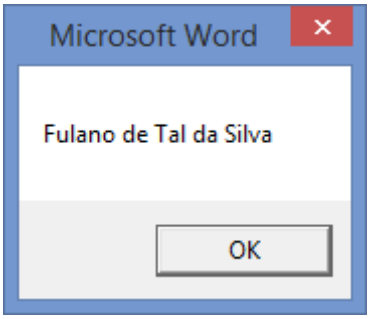
Exemplo:

```
Sub entradadedados()
  Dim nom As String
     nom = InputBox("Digite o seu nome.", "Usuário")
  MsgBox nom
End Sub
```

Ao executarmos a macro **entradadedados** será solicitado ao usuário que digite o seu nome e em seguida o mesmo é exibido em uma caixa de mensagem.



Ao clicar no botão OK será exibido o nome em uma caixa de mensagem.

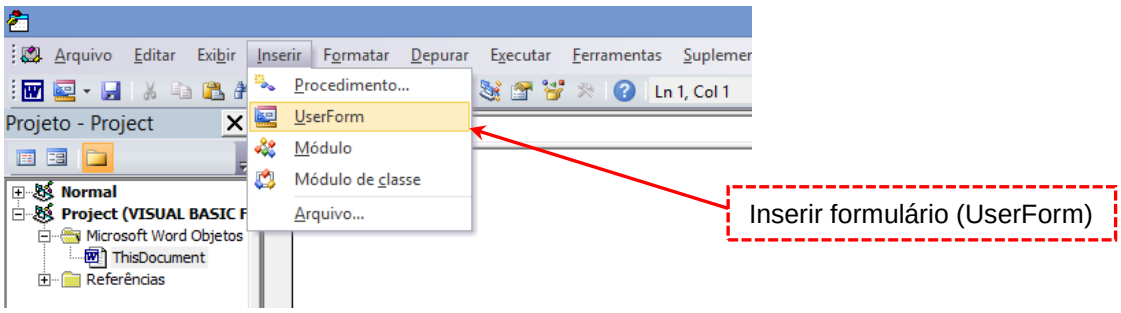


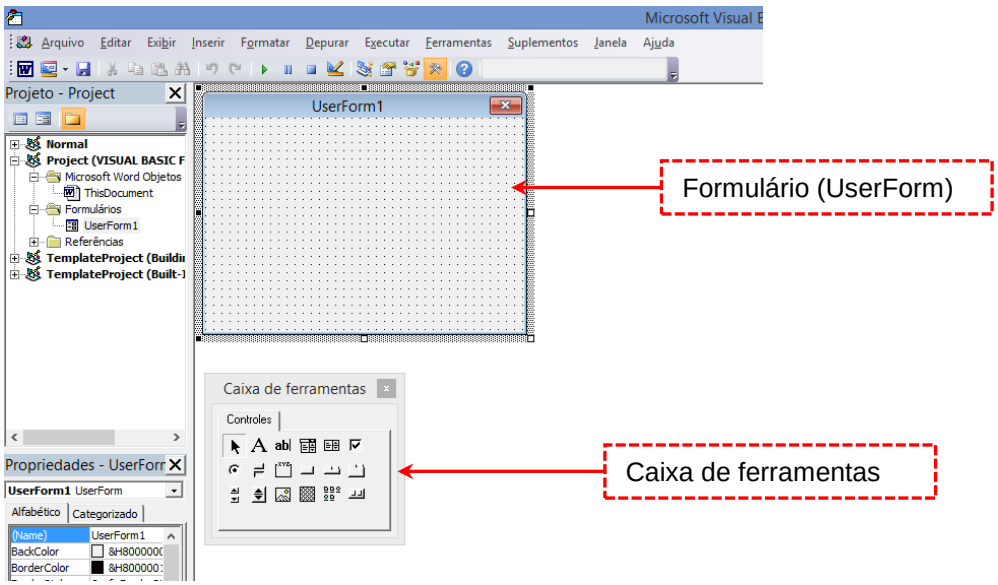
2.3-Criando Formulários

Como neste capítulo estamos falando de interface, não poderíamos prosseguir sem comentar sobre a forma mais interessante de se criar interfaces no VBA que é por meio de formulários.
Um formulário funciona como caixas de mensagem com controles para coletar dados do usuário, enviar dados, mostrar dados, mostrar opções, etc.

Como vimos no Capítulo I, quando inserimos um formulário no ambiente de edição do VBA a caixa de ferramentas é automaticamente exibida. Basta clicar no controle desejado e soltá-lo sobre o formulário para criarmos a interface desejada. Na janela Propriedades podemos alterar o formulário e os controles nele inseridos para deixarmos com o melhor visual possível.

Inserindo um Formulário.





Como foi dito no início desta apostila, o primeiro e o segundo capítulos são apenas uma breve introdução ao VBA. Poderíamos até descrever detalhadamente todas as outras ferramentas do editor, poderíamos nos aprofundar no escopo de variáveis e constantes, nos métodos e propriedades, nas estruturas de programação (estrutura condicional IF, estrutura SELECT CASE, controles de LOOPING, etc), mas acho que é bem mais útil descrever as bases do funcionamento e dar os sedimentos para o leitor ter condições suficientes para seguir sozinho. Além do mais, nos capítulos seguintes apresentarei vários exemplos de macros e soluções que os usuários do Word poderão implementá-los em seu dia a dia e também estarei mostrando passo a passo como criar 10 Projetos utilizando uma linguagem simples, sem os termos técnicos, de maneira que aqueles que são leigos no assunto possam compreender toda lógica envolvida e partir daí se aventurar a criar suas próprias aplicações.

CAPÍTULO - III

MACROS

3.-O QUE É MACRO?

Uma macro nada mais é que uma sequência de comandos e funções armazenados em um módulo de VBA e que pode ser usada sempre que você precisar executar a tarefa, portanto, macros são excelentes recursos para realizar e automatizar tarefas repetitivas e longas.

Muitos tem uma certa "repulsa" ao falarmos em macros, pois envolve a programação VBA (Visual Basic for Applications), porém, o que muita gente não sabe é que o Word possui um gravador de macros para automatizar esta programação, de maneira que não é necessário conhecer a linguagem VBA para criar uma macro no Word, pois, ao utilizar este mecanismo ele grava a sequencia de todas as ações enquanto o gravador estiver ligado.

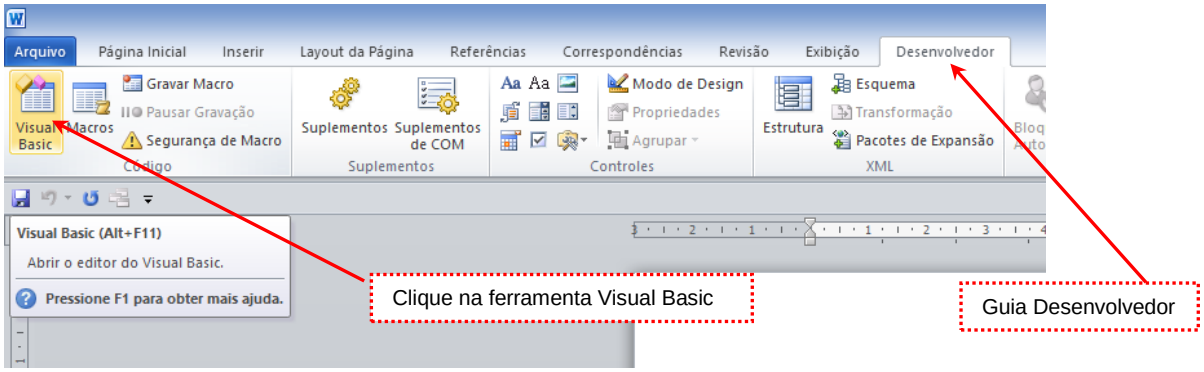
Apesar do gravador de macros ser uma mão na roda para quem quer automatizar tarefas repetitivas, com o tempo torna-se imprescindível aprender e dominar a linguagem VBA para criar macros mais sofisticadas sem a necessidade de gravá-las passo a passo.

A edição a partir dos códigos é feita através do próprio Editor do Visual Basic contido no programa e esta ferramenta que utilizaremos para criar as 15 (quinze) macros que apresentaremos neste capítulo.

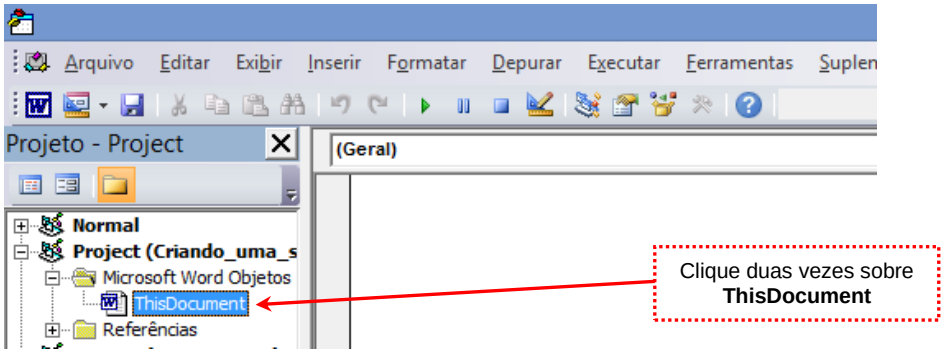
Mas cuidado, preste muita atenção ao nomear uma macro, pois, se você der a uma nova macro o mesmo nome que uma interna do Word, as ações da nova macro substituirão a macro interna. Para exibir uma lista de macros internas, acione a guia **Desenvolvedor** e no grupo **Código** clique em **Macros**. Na lista **Macros em**, clique em **Comandos do Word**.

3.1-Criando uma saudação ao usuário

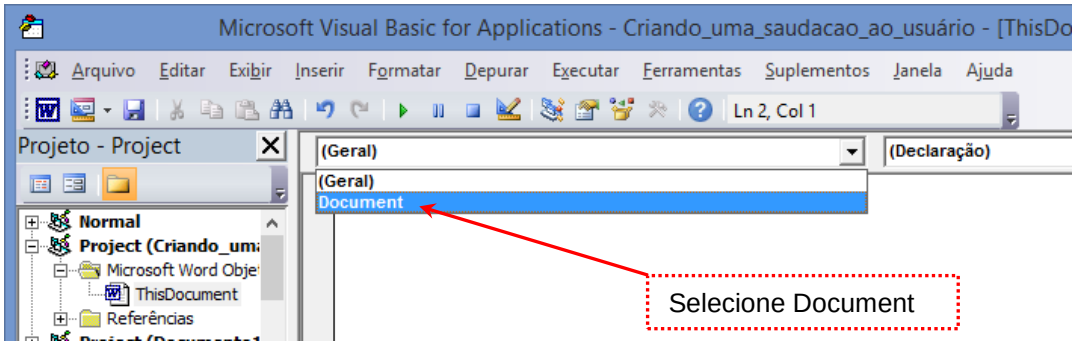
Abra um documento Word existente ou um novo documento. Abra o Editor do Visual Basic (VBE) através da **Guia Desenvolvedor** e no **Grupo Código** clique em **Visual Basic** ou simplesmente através do atalho **Alt + F11**.



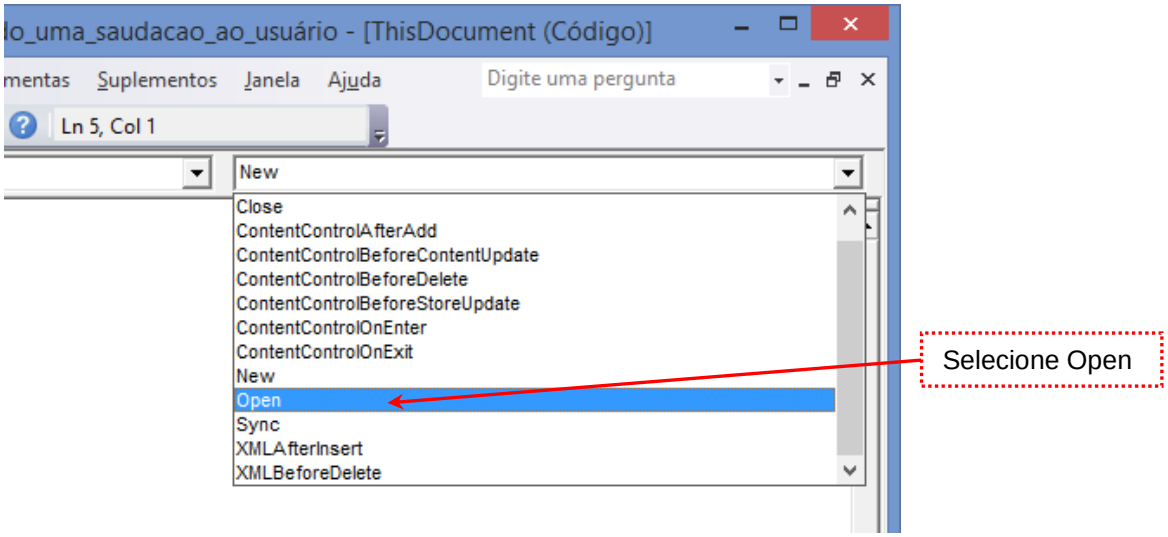
No ambiente do VBA clique duas vezes sobre "**ThisDocument**" que fica na janela **Projeto –Project**.



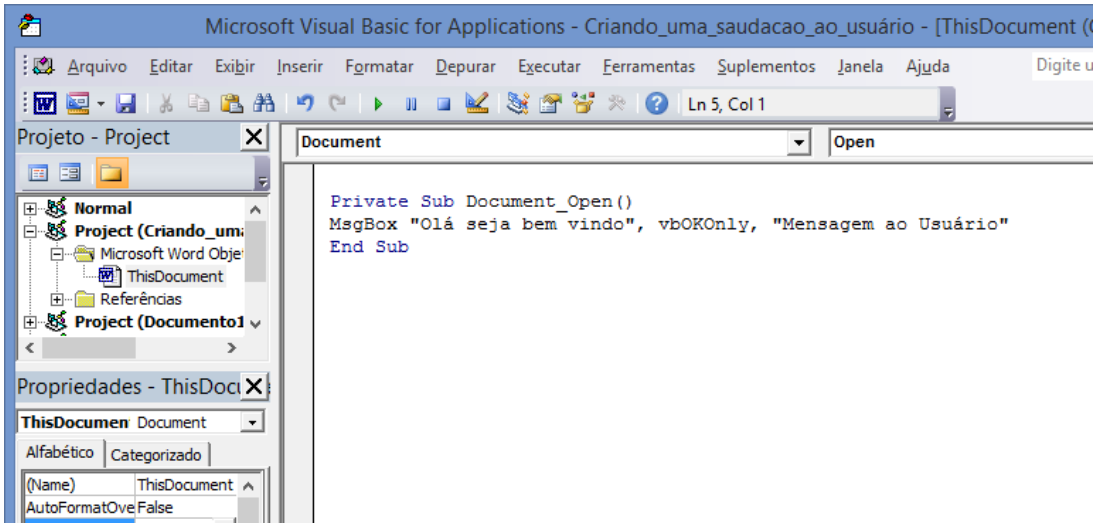
Na **Janela de Código** selecione o objeto **Document**:



Em procedimento selecione **Open**:



Veja como deve ficar na imagem abaixo:

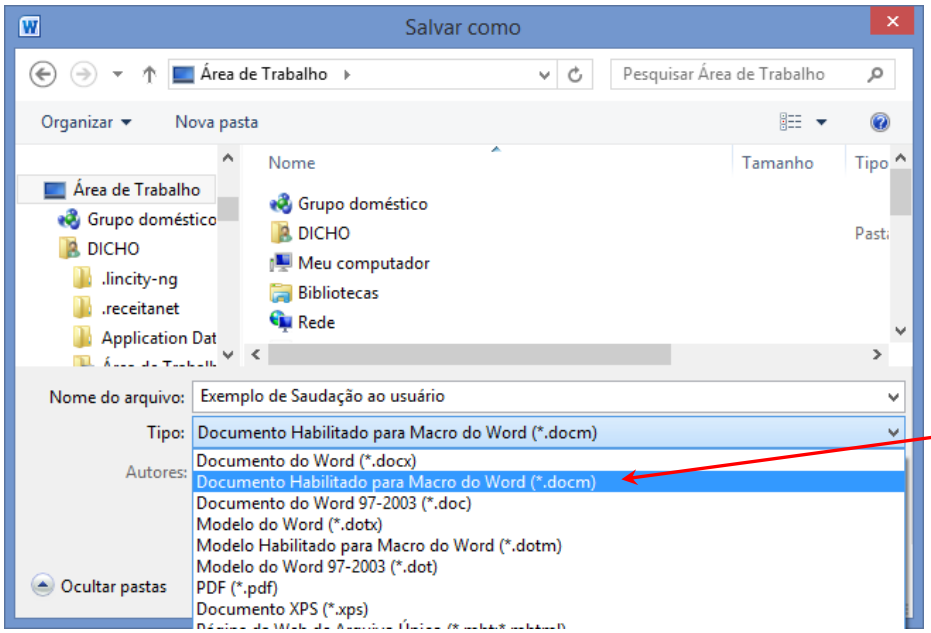


Explicando o Código:

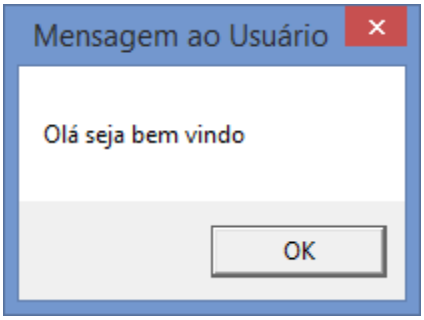
```
Private Sub Document_Open()
MsgBox "Olá seja bem vindo", vbOKOnly, "Mensagem ao Usuário"
End Sub
```

Observe que para este exemplo utilizamos o procedimento **Open**, isso significa que toda vez que o documento for aberto será exibo uma mensagem de Boas vindas ao usuário.

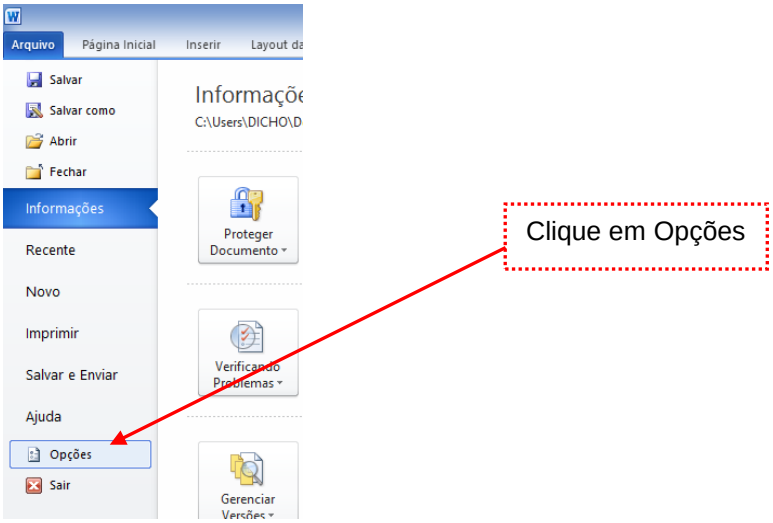
Feche o Editor do Visual Basic e salve o documento como **Documento Habilitado para Macro do Word**.



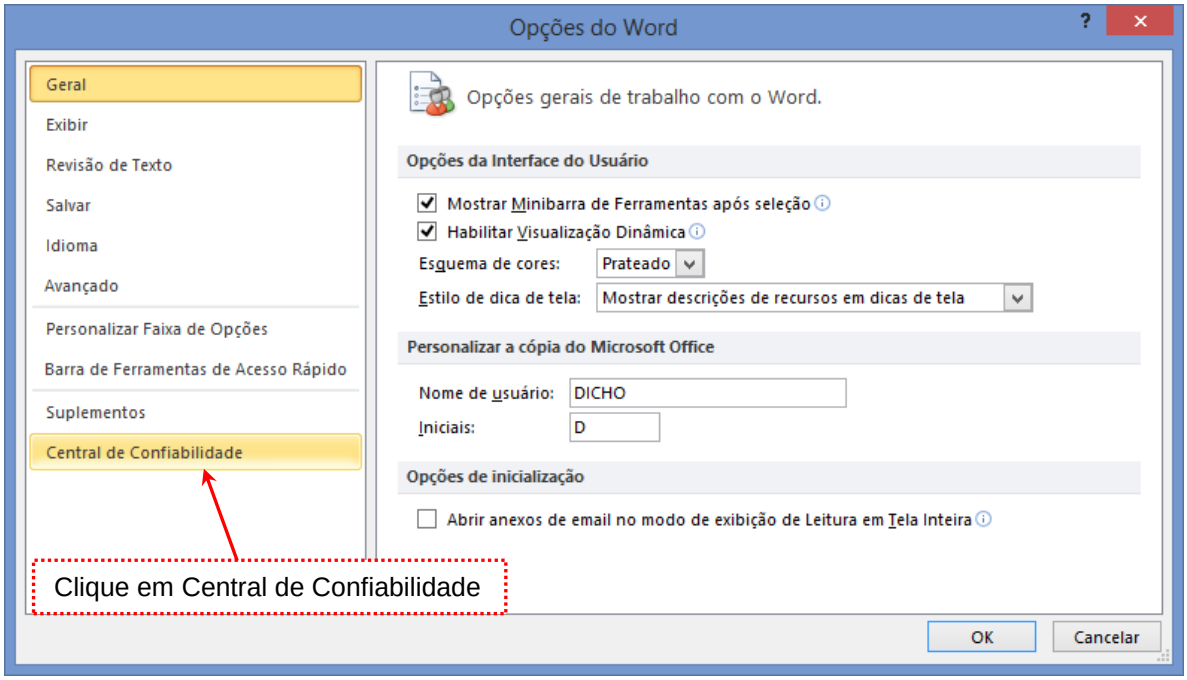
Feche o documento e em seguida abra-o novamente para ver o resultado:



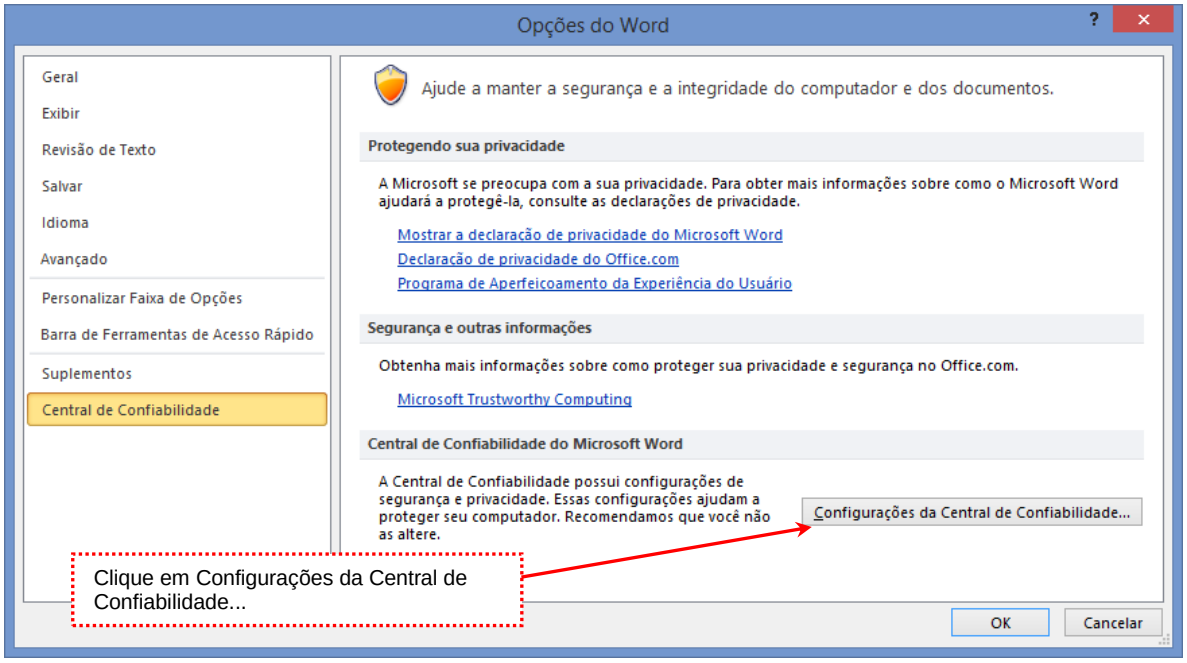
Observação importante: Para que uma macro possa ser executada é necessário alterar a configuração de Macro do Word através da Guia **Arquivo** >> **Opções** >> **Central de Confiabilidade**. Como mostra as imagens abaixo:



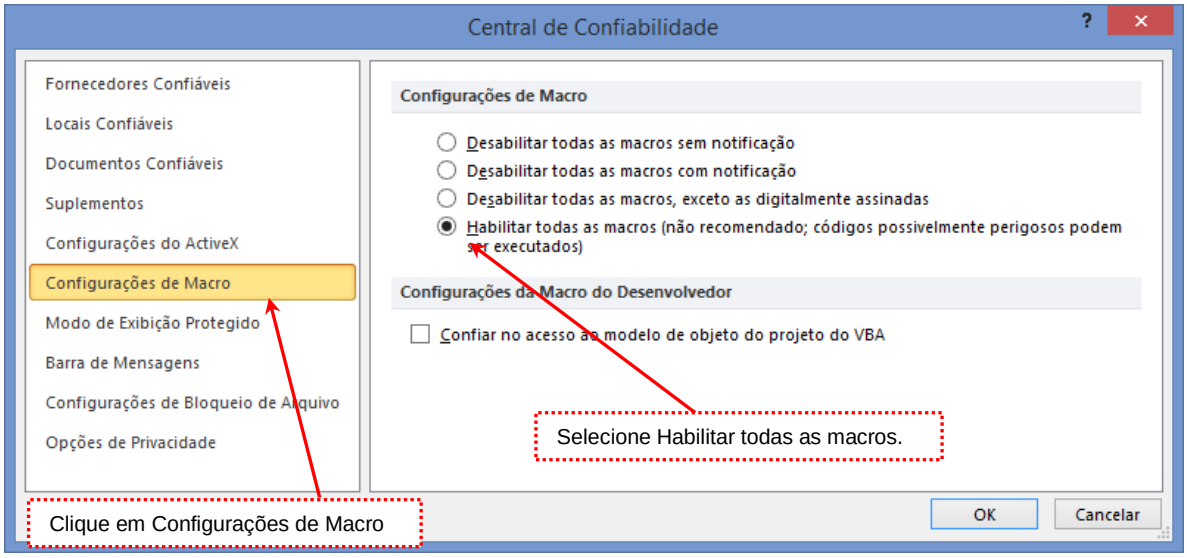
Na janela de Opções do Word clique em **Central de Confiabilidade**:



Clique em **Configurações da Central de Confiabilidade...**

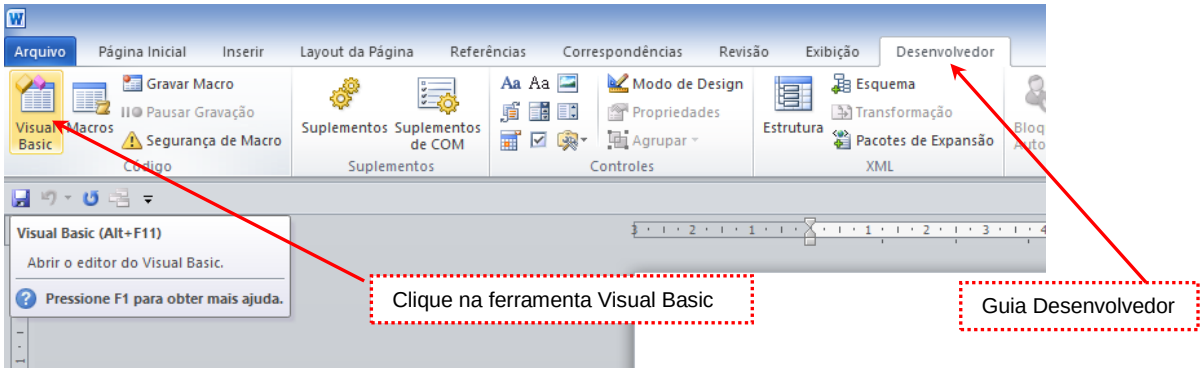


Em Central de Confiabilidade clique em **Configurações de Macro** e em Configurações de Macro selecione a opção **Habilitar todas as macros**.

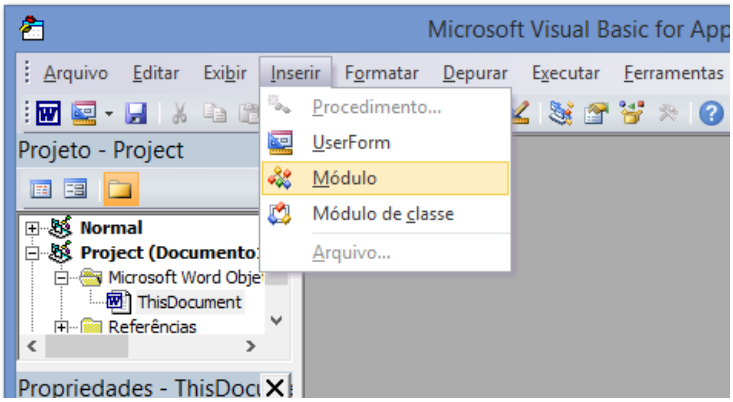


3.2-Exibindo uma saudação ao usuário de acordo com o horário

Abra um documento Word existente ou um novo documento. Abra o Editor do Visual Basic através da **Guia Desenvolvedor** e no **Grupo Código** clique em **Visual Basic** ou simplesmente através do atalho **Alt + F11**.



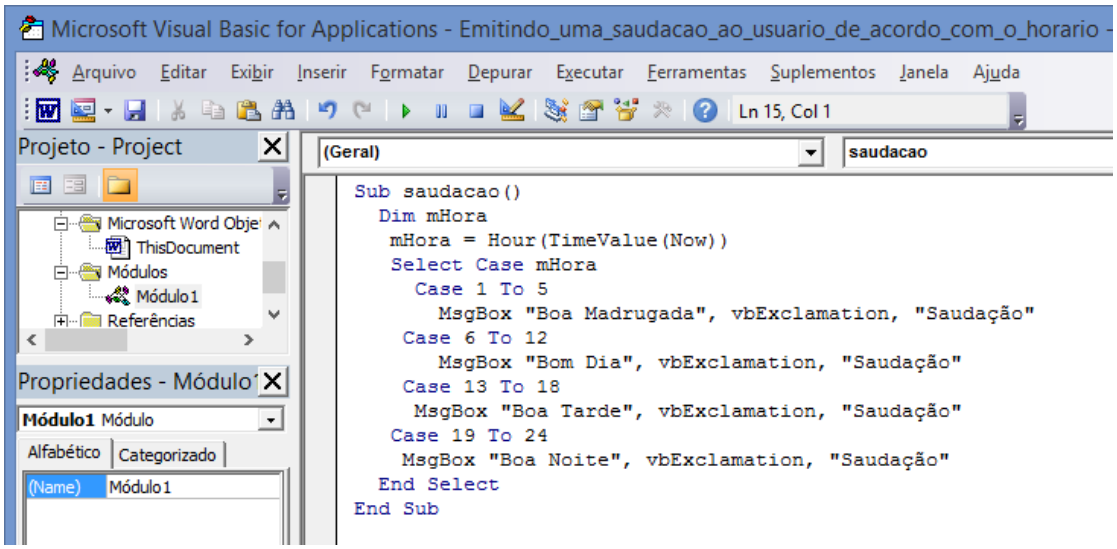
No ambiente de programação do VBA clique no menu **Inserir >> Módulo** para abrir a janela de edição dos códigos (**Módulo1**).



Digite o código abaixo:

```
Sub saudacao()
  Dim mHora
   mHora = Hour(TimeValue(Now))
   Select Case mHora
     Case 1 To 5
       MsgBox "Boa Madrugada", vbExclamation, "Saudação"
    Case 6 To 12
       MsgBox "Bom Dia", vbExclamation, "Saudação”
    Case 13 To 18
     MsgBox "Boa Tarde", vbExclamation, "Saudação"
   Case 19 To 24
    MsgBox "Boa Noite", vbExclamation, "Saudação"
  End Select
End Sub
```

Observe a imagem:



Explicando o Código:

As linhas de código acima cria uma macro chamada **"saudacao"** cuja rotina emitirá em uma caixa de mensagem, uma saudação ao usuário, de acordo com o horário.

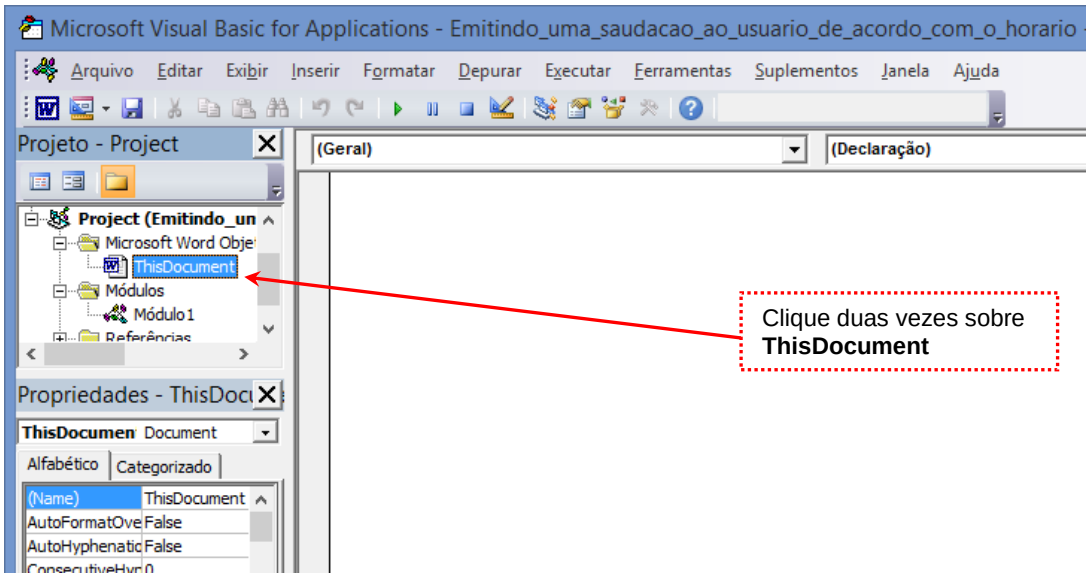
Primeiramente declaramos a varíavel (aqui neste exemplo demos o nome de mHora);

Utilizamos a função Hour que retorna como parâmetro uma hora, e a função TimeValue juntamente com a propriedade Now que retorna a hora atual do sistema.

A estrutura Select Case testa um valor dentre os vários possíveis e determina qual Grupo de código será chamado:

- Se o documento for aberto no horário das 01:00 h às 05:00 h será exibido a mensagem: "Boa Madrugada".
- Se o documento for aberto no horário das 06:00 h às 12:00 h será exibido a mensagem: "Bom Dia".
- Se o documento for aberto no horário das 13:00 h às 18:00 h será exibido a mensagem: "Boa Tarde".
- Se o documento for aberto no horário das 19:00 h às 24:00 h será exibido a mensagem: "Boa Noite"

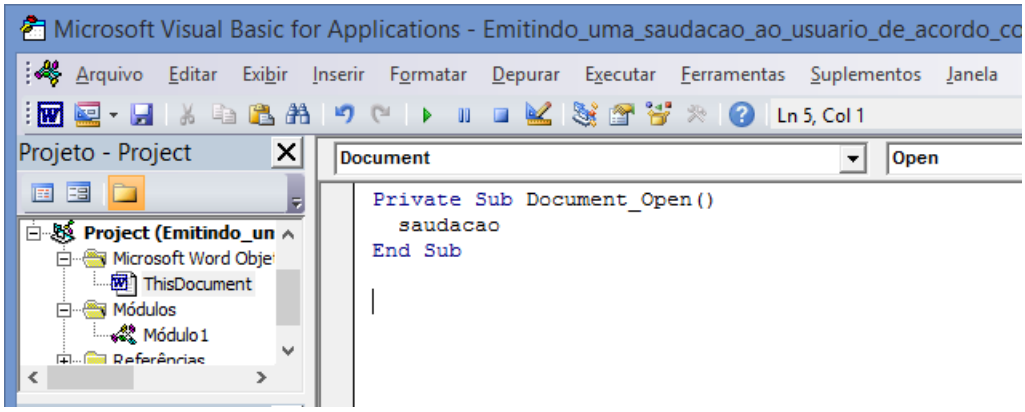
Agora clique duas vezes sobre **“ThisDocument”** que fica na janela Projeto – Project.



Insira o seguinte código:

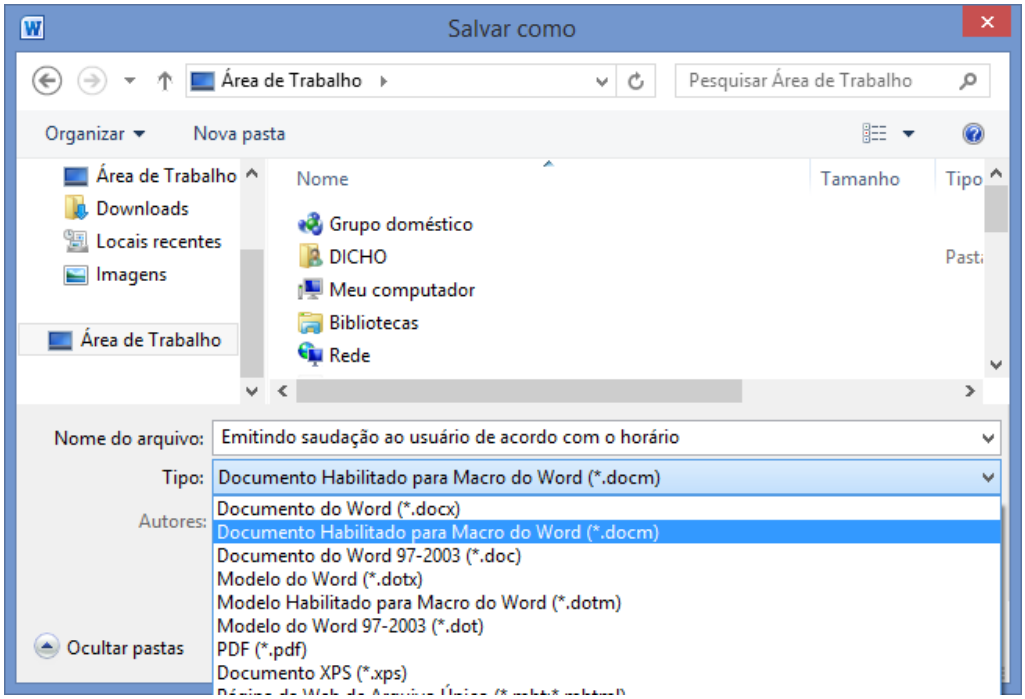
```
Private Sub Document_Open()
  saudacao
End Sub
```

Veja a imagem abaixo:



Estas linhas de código fará com que a macro **“saudacao”** seja executada todas as vezes em que o documento for aberto.

Feche o Editor do Visual Basic e salve o documento como **Documento Habilitado para Macro do Word**.

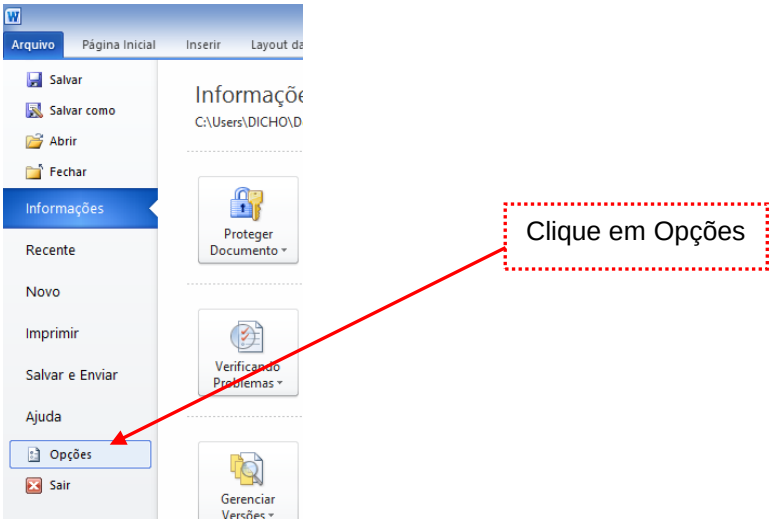


Feche o documento e abra-o novamente. Será exibido uma mensagem ao usuário de acordo com o horário em que o documento está sendo aberto

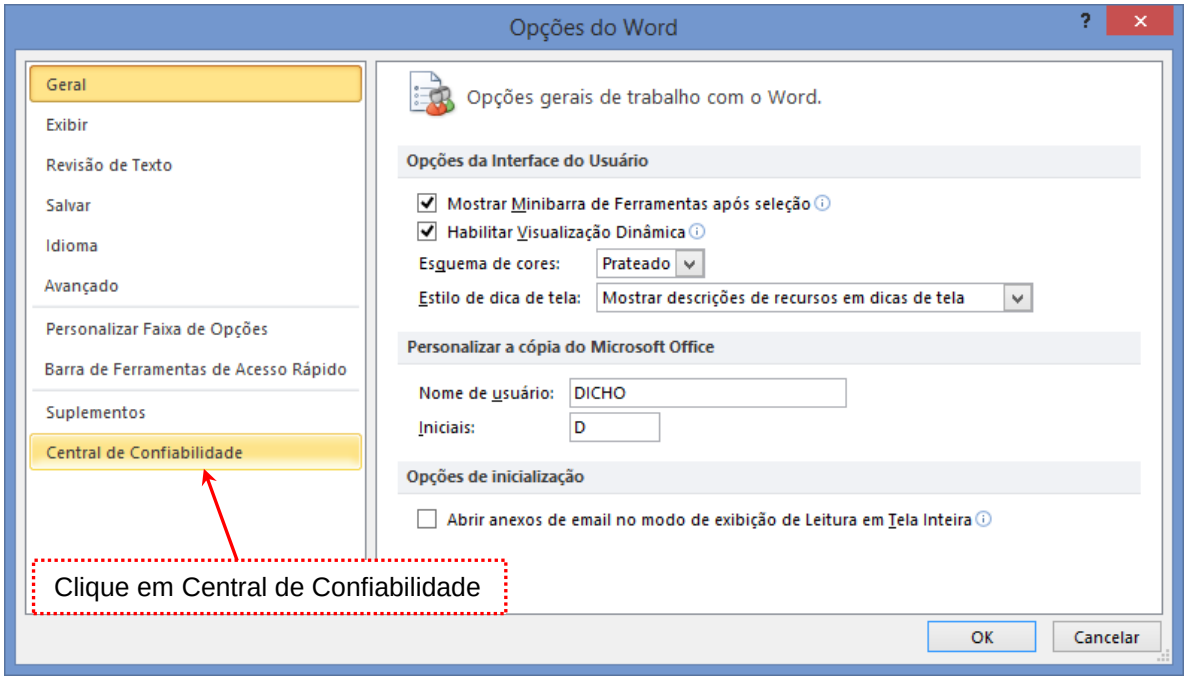
Veja o resuldado:



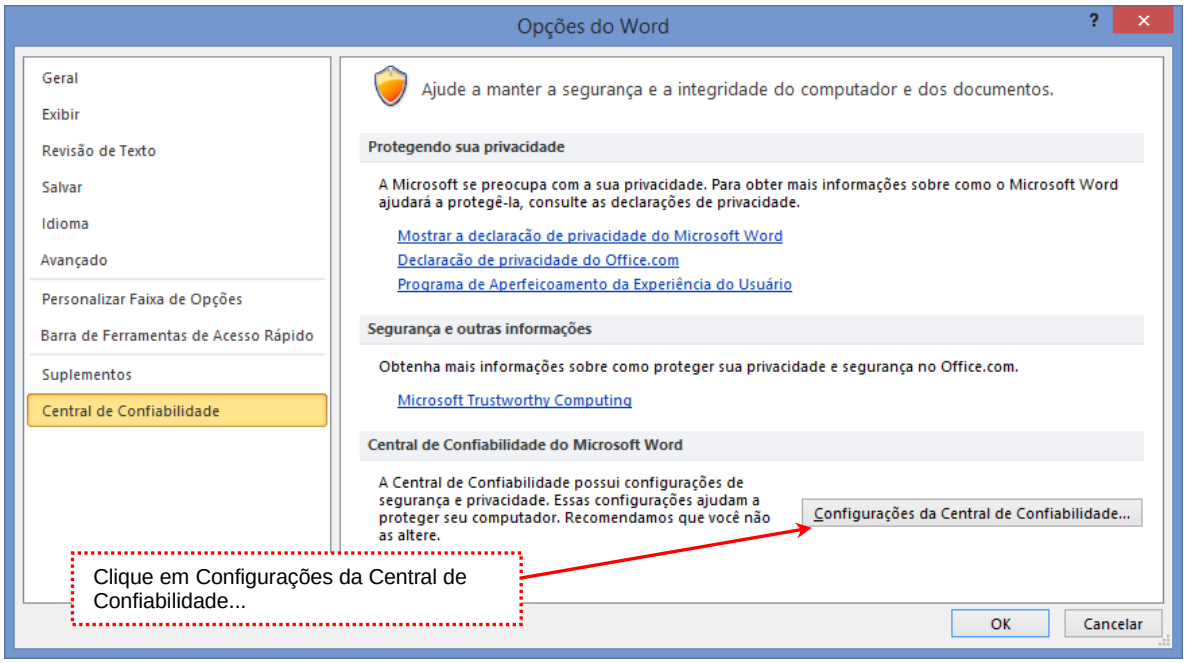
Observação importante: Para que uma macro possa ser executada é necessário alterar a configuração de Macro do Word através da Guia **Arquivo** >> **Opções** >> **Central de Confiabilidade**. Como mostra as imagens abaixo:



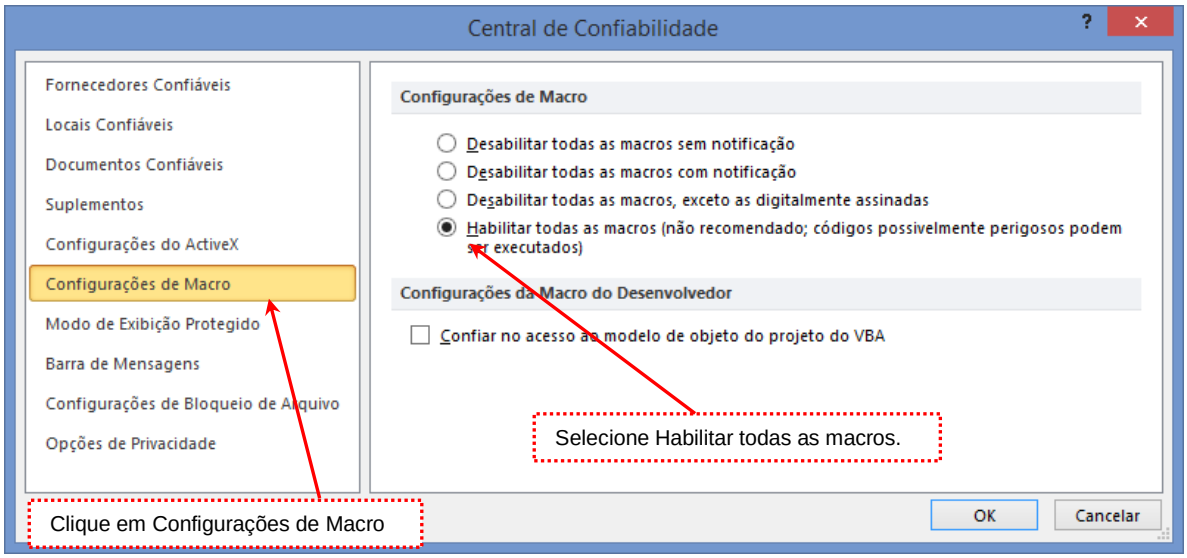
Na janela de Opções do Word clique em **Central de Confiabilidade**:



Clique em **Configurações da Central de Confiabilidade...**



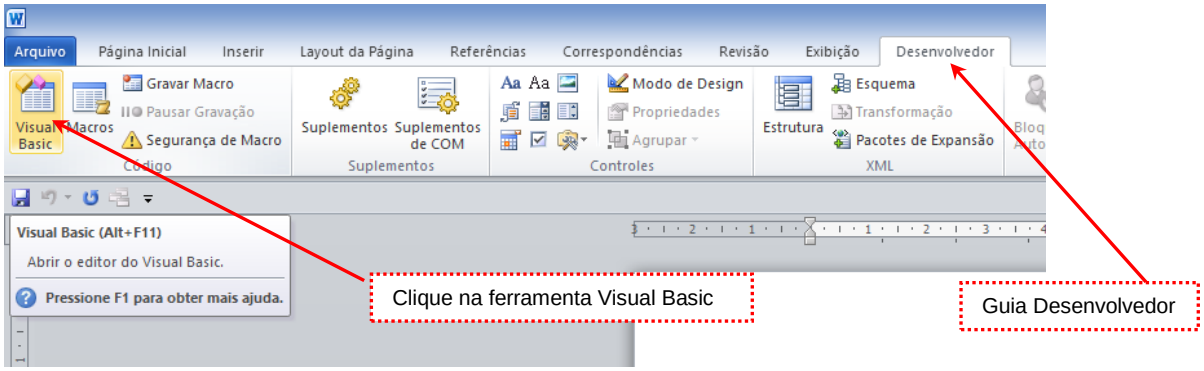
Em Central de Confiabilidade clique em **Configurações de Macro** e em Configurações de Macro selecione a opção **Habilitar todas as macros**.



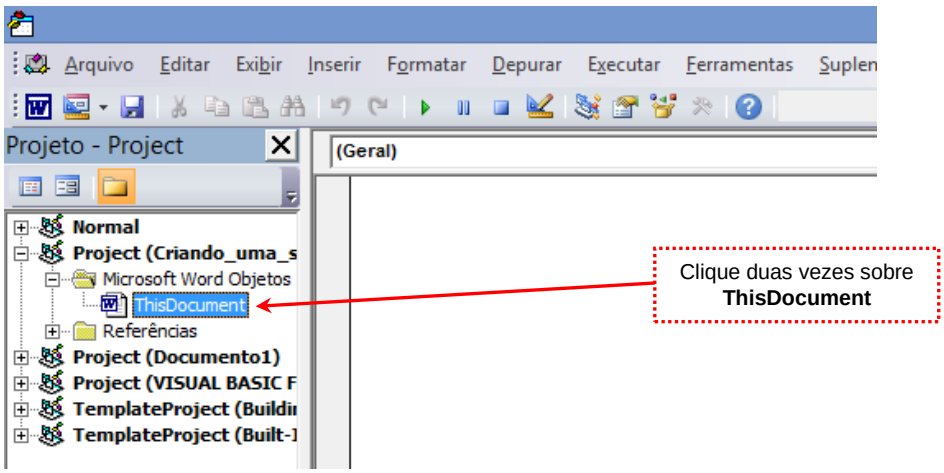
3.3-Saudação – interagindo com usuário

Outra forma simples de saudação ao usuário:

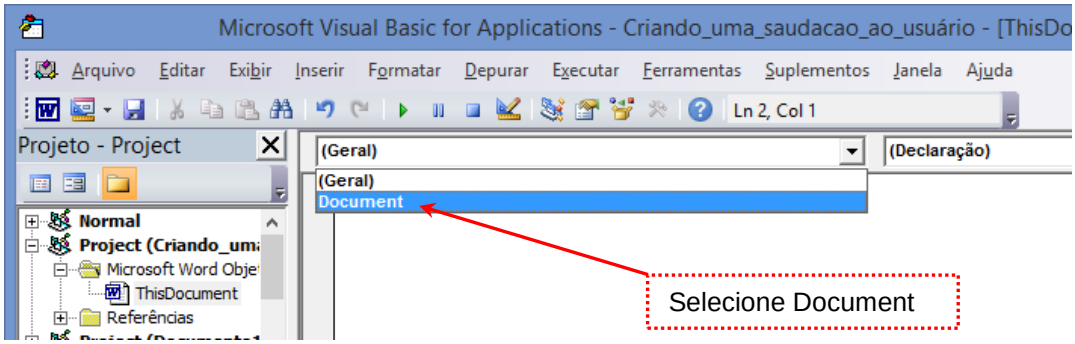
Abra um documento Word existente ou um novo documento. Abra o Editor do Visual Basic (VBE) através da **Guia Desenvolvedor** e no **Grupo Código** clique em **Visual Basic** ou simplesmente através do atalho **Alt + F11**.



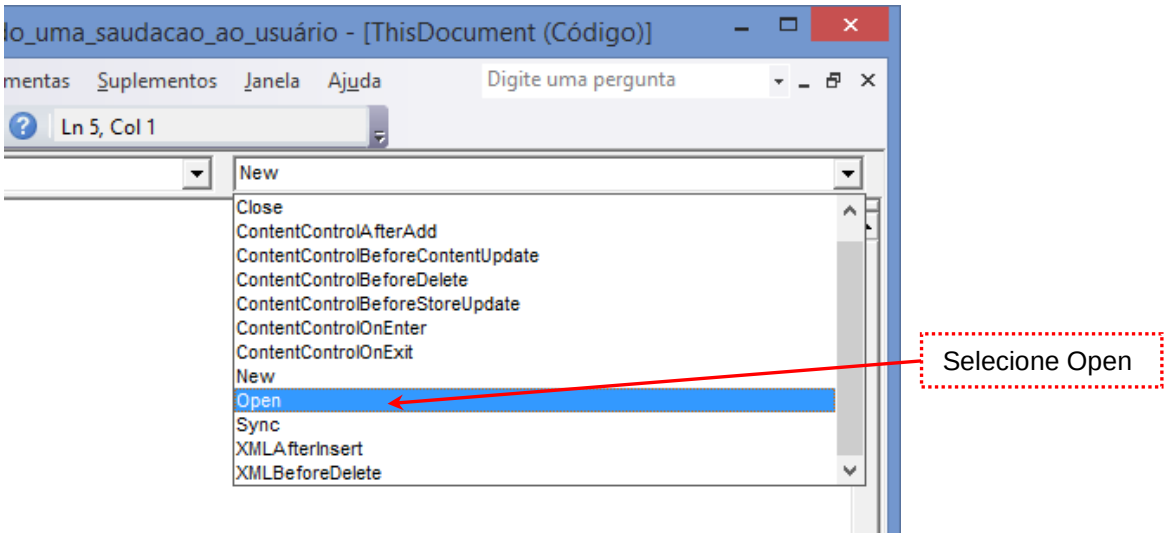
No ambiente do VBA clique duas vezes sobre "**ThisDocument**" que fica na janela **Projeto –Project**.



Na **Janela de Código** selecione o objeto **Document**:



Em procedimento selecione **Open**:



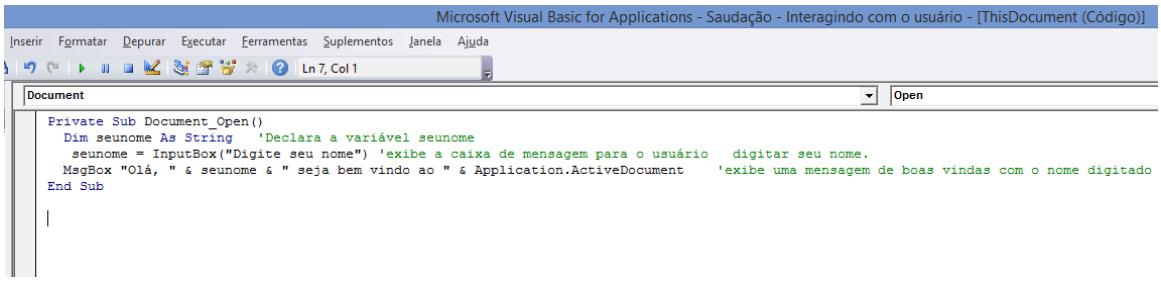
Digite o código abaixo:

```
Private Sub Document_Open()
Dim seunome As String       'Declara a variável seunome
seunome = InputBox("Digite seu nome")    'exibe a caixa de mensagem para o usuário   digitar seu nome.
 MsgBox "Olá, " & seunome & " seja bem vindo ao " & Application.ActiveDocument    'exibe uma
mensagem de boas vindas com o nome digitado InputBox exibindo também o nome do documento ativo.
End Sub
```

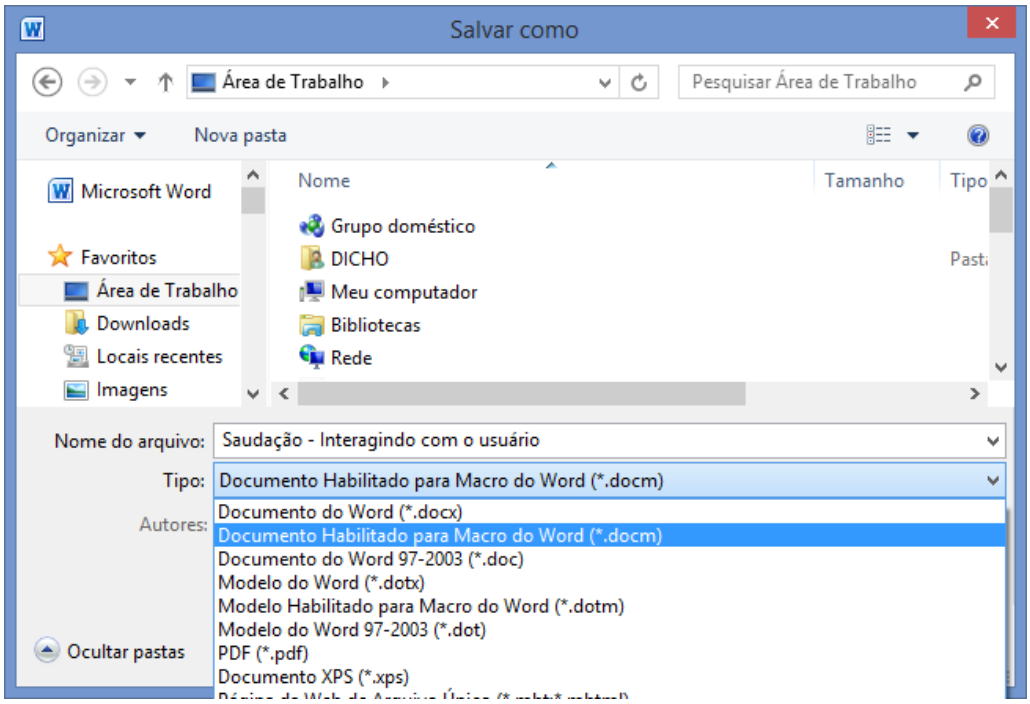
Explicando o Código:

O valor da variável **seunome** é igual ao nome que foi digitado na InputBox. Ao clicar no Botão OK será exibida uma caixa de mensagem de boas vindas com o nome do usuário e o nome do Documento ativo.

OBS: As frases em verde são apenas comentários que foram inseridos para melhor entender o código. Não deve ser tirado o **'** (apóstrofo) porque se for tirado deixará de ser um comentário e o editor entenderá como linha de código o que gerará erros na sua execução.

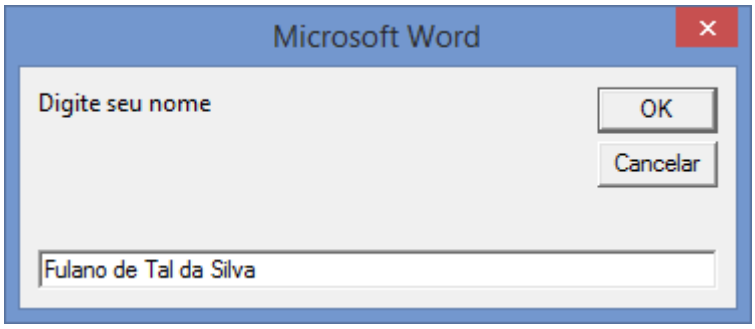


Salve o documento como **Documento Habilitado para Macro do Word**.

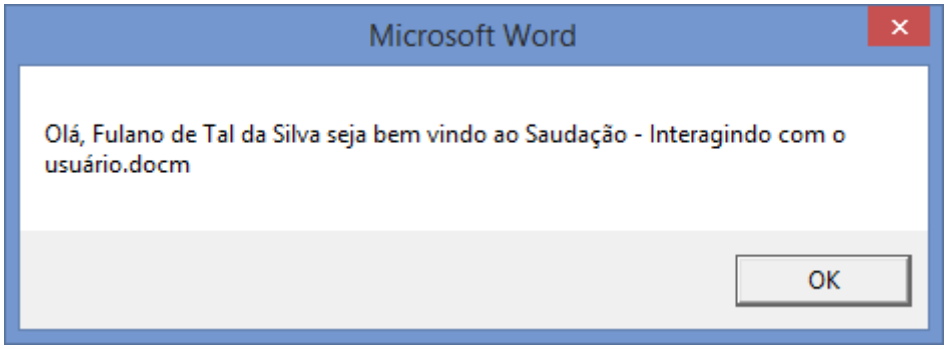


Feche o documento e em seguida abra-o novamente para ver o resultado.

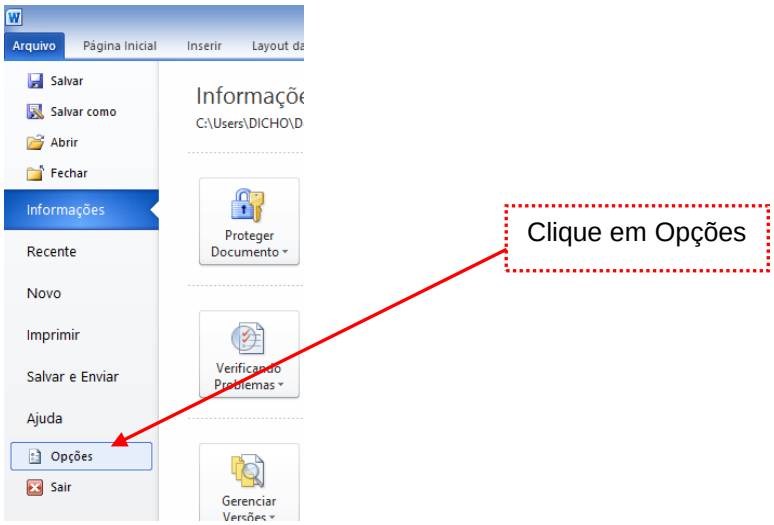
Será exibida uma caixa de mensagem para que o usuário insira o seu nome.



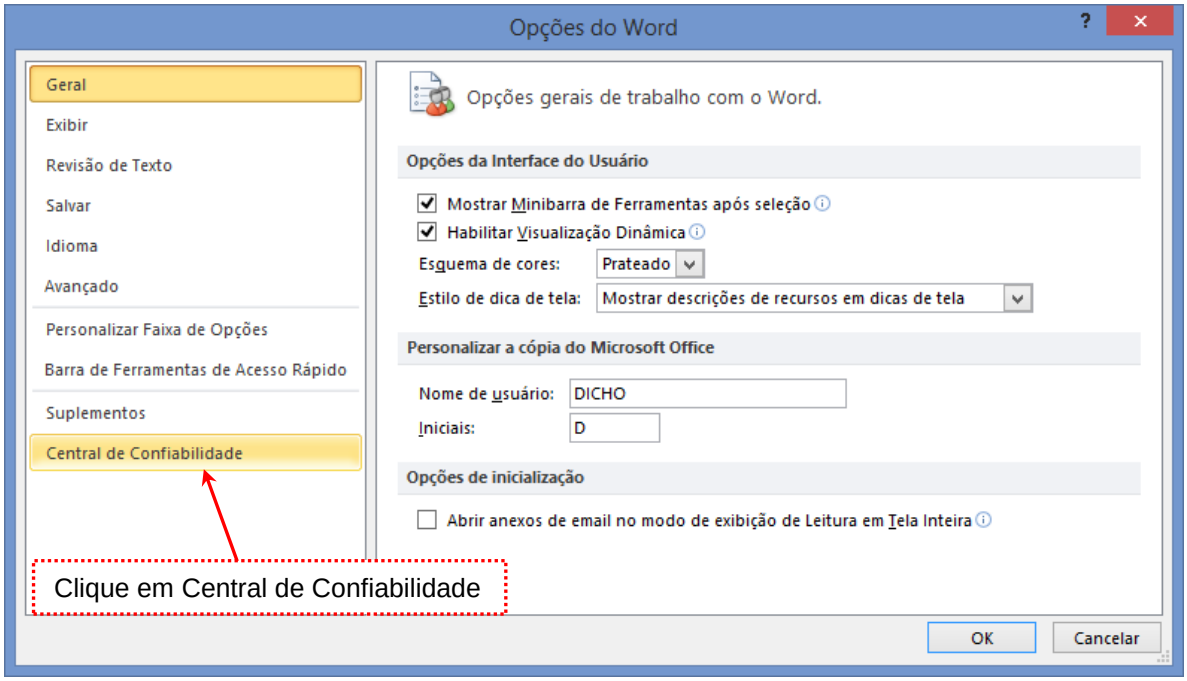
Após inserir o nome e clicar no botão OK será exibida outra caixa de mensagem com o nome do usuário e o nome do documento.



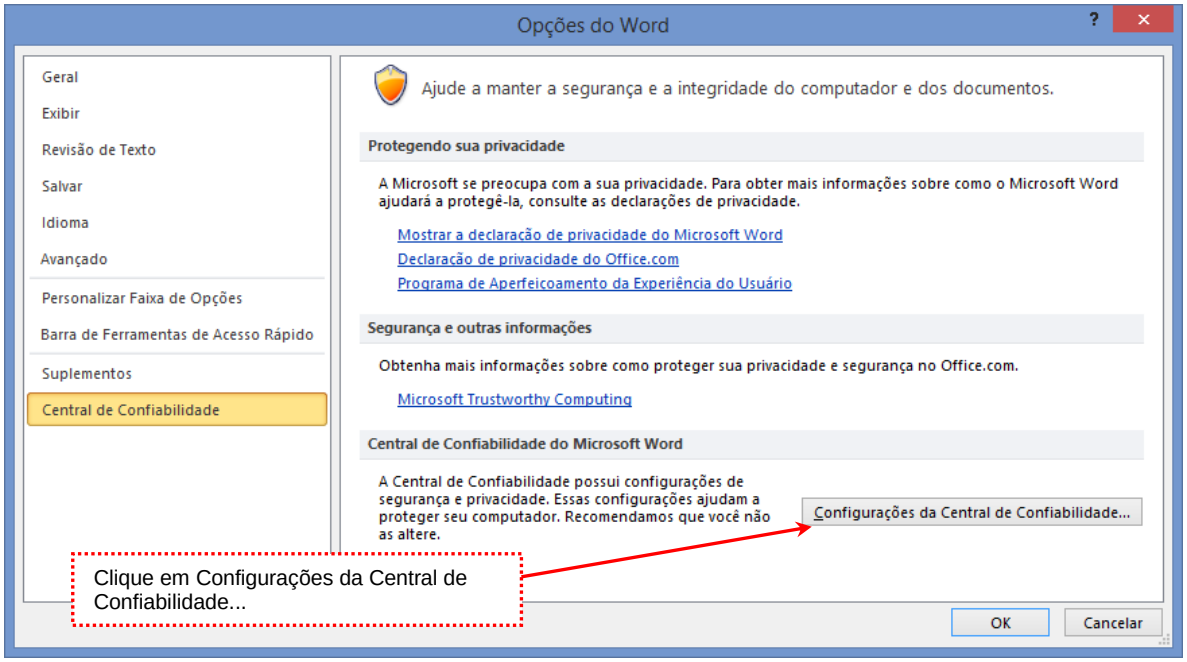
Observação importante: Para que uma macro possa ser executada é necessário alterar a configuração de Macro do Word através da Guia **Arquivo** >> **Opções** >> **Central de Confiabilidade**. Como mostra as imagens abaixo:



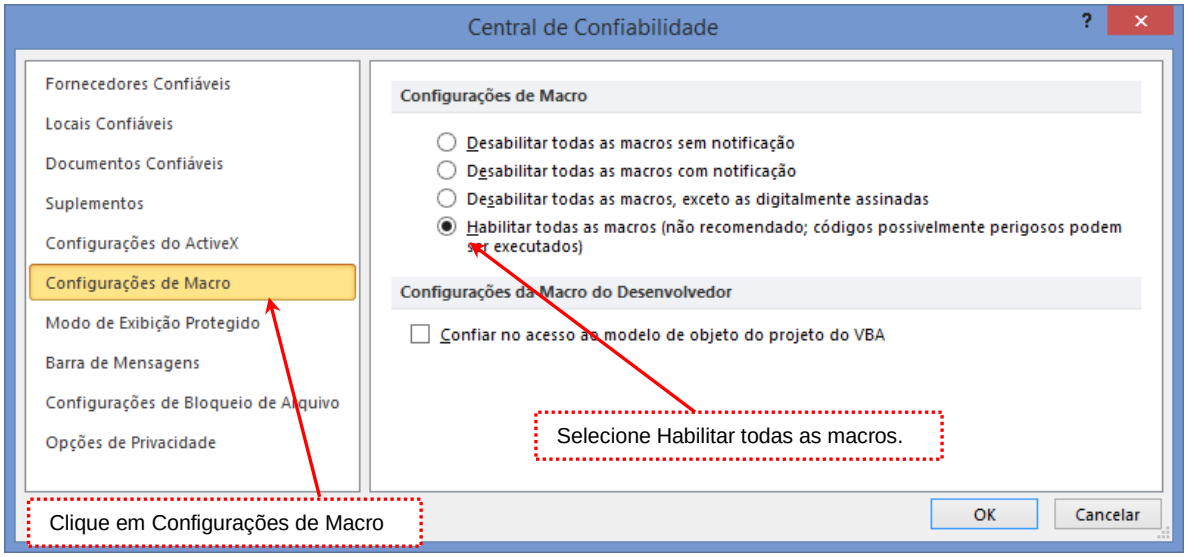
Na janela de Opções do Word clique em **Central de Confiabilidade**:



Clique em **Configurações da Central de Confiabilidade...**



Em Central de Confiabilidade clique em **Configurações de Macro** e em Configurações de Macro selecione a opção **Habilitar todas as macros**.

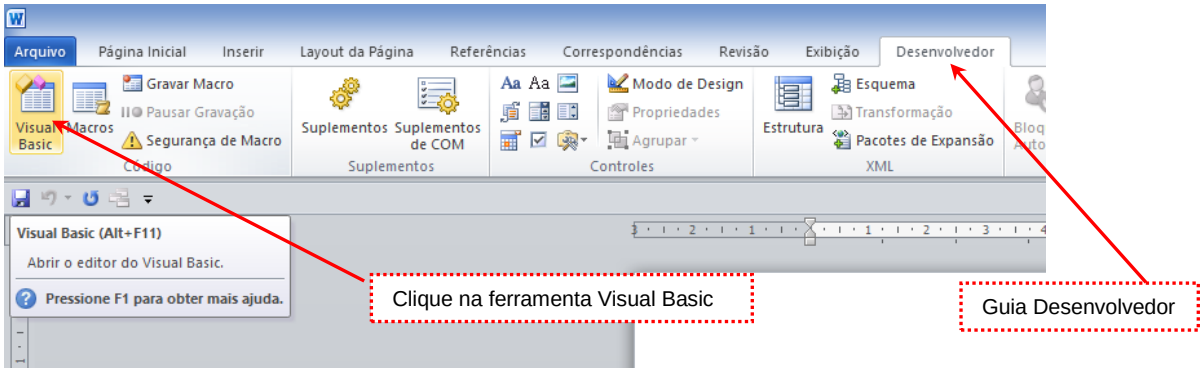


3.4-Outro exemplo utilizando a função INPUTBOX

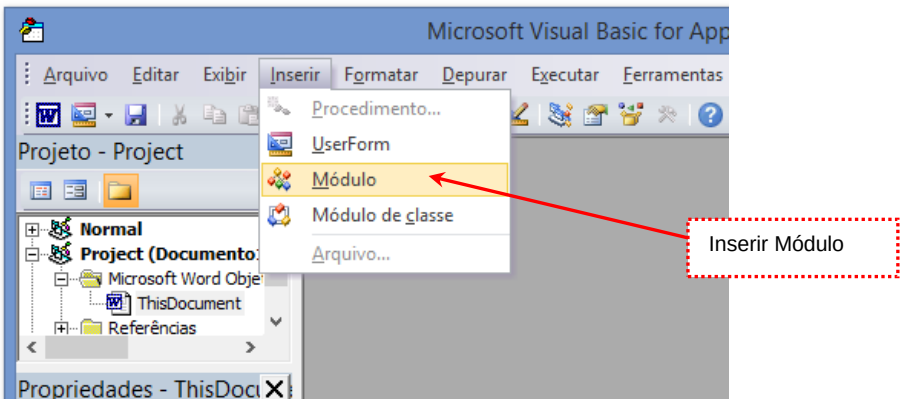
Outro exemplo utilizando a função INPUTBOX para calcular valores e apresentar o resultado em uma caixa de mensagem.

Neste exemplo usaremos duas InputBox, para realizar a soma de valores.

Abra um documento Word existente ou um novo documento. Abra o Editor do Visual Basic através da **Guia Desenvolvedor** e no **Grupo Código** clique em **Visual Basic** ou simplesmente através do atalho **Alt + F11**.



No ambiente de programação do VBA clique no menu **Inserir >> Módulo** para abrir a janela de edição dos códigos (**Módulo1**) e crie a seguinte macro com o nome calculavalores:

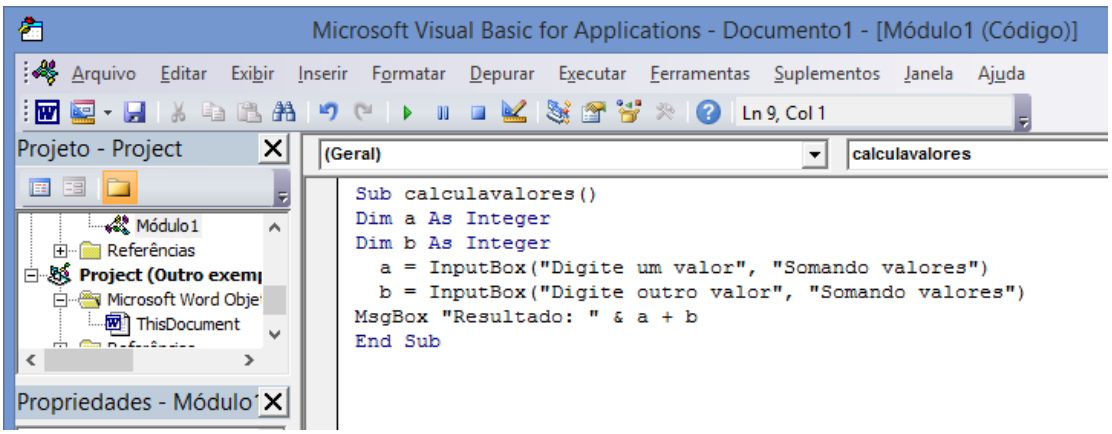


Na janela de Código digite:

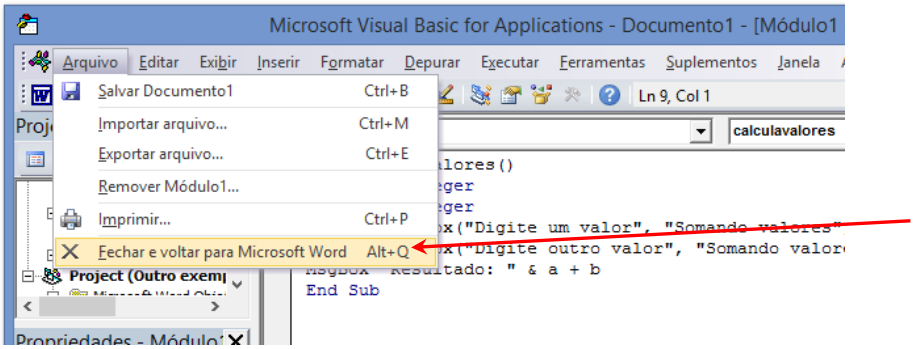
```
Sub calculavalores()
Dim a As Integer
Dim b As Integer
  a = InputBox("Digite um valor", "Somando valores")
  b = InputBox("Digite outro valor", "Somando valores")
MsgBox  “Resultado= “ &  a + b
End Sub
```

Explicando o Código:

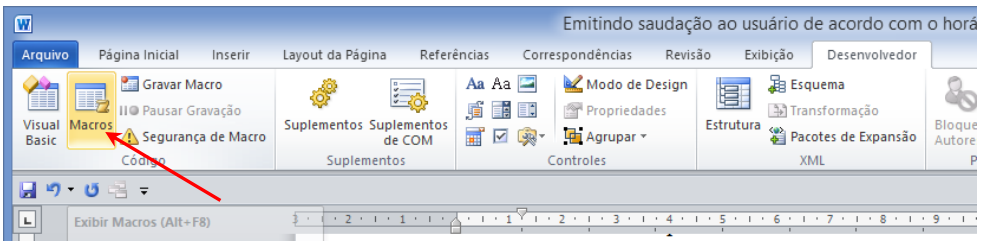
Primeiramente declaramos duas variáveis “a” e “b” como tipo Integer.
O valor da variável “a” é igual ao valor digitado na primeira InputBox e o valor da variável “b” é igual ao valor digitado na segunda InputBox.
Será exibida uma caixa de mensagem com a soma dos valores de “a” e “b”.



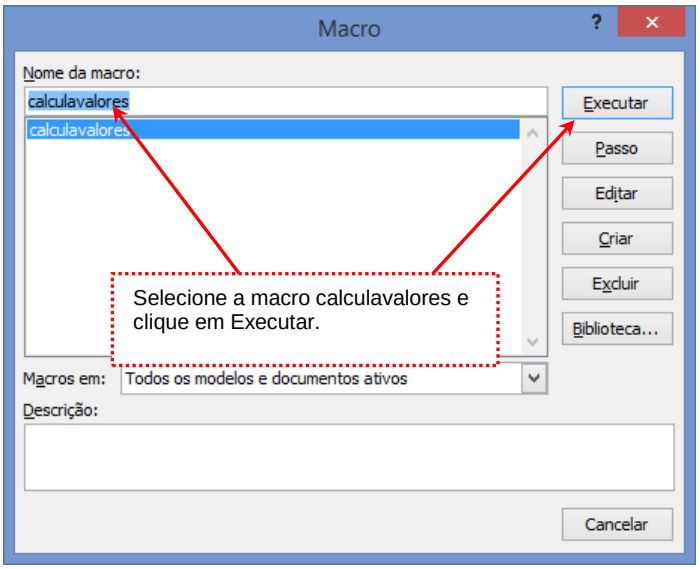
Agora feche o Editor do Visual Basic para voltar ao Word. Clique no Menu **Arquivo** e em seguida em **Fechar e voltar para Microsoft Word**.



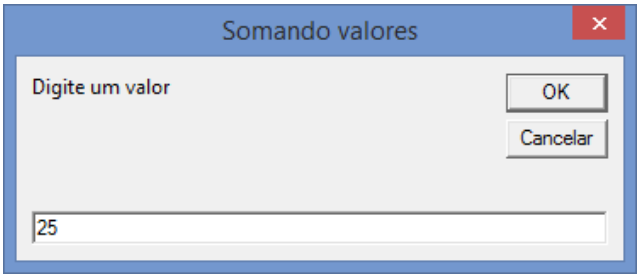
Acione a Guia **Desenvolvedor** e no Grupo **Código** clique em **Macros** ou se preferir use a combinação de teclas (teclas de atalho) **Alt + F8.**



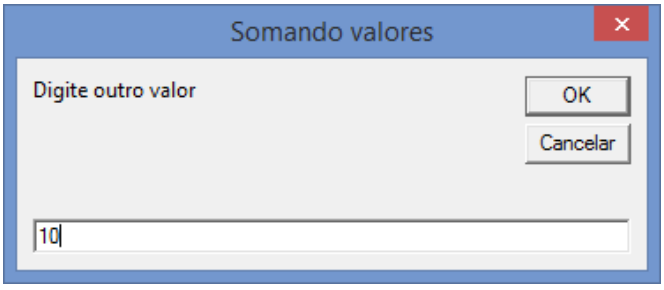
Selecione a macro **calculavalores** e clique em **Executar**.



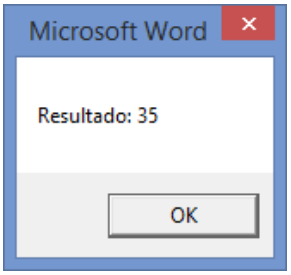
Surgirá uma caixa de mensagem para que seja inserido o primeiro valor.



Insira um valor e e clique no botão OK. Surgirá uma nova caixa de mensagem para que seja inserido o segundo valor.



Insira o segundo valor e e clique no botão OK. O resultado da soma será exibida em uma nova caixa de mensagem.

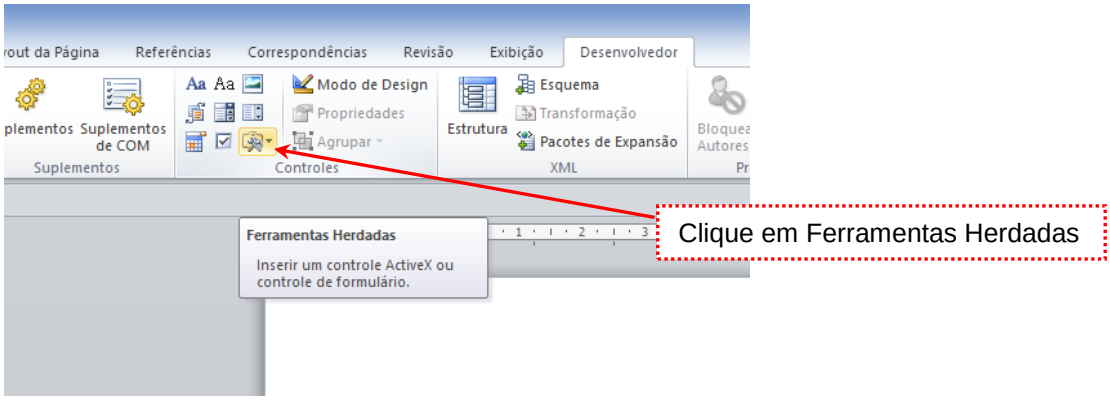


Podemos ainda associar essa macro a um botão usando um **Controle ActiveX** ou adicioná-la a **Faixa de Opções** ou ainda inseri-la na **Barra de Ferramentas de Acesso Rápido**.

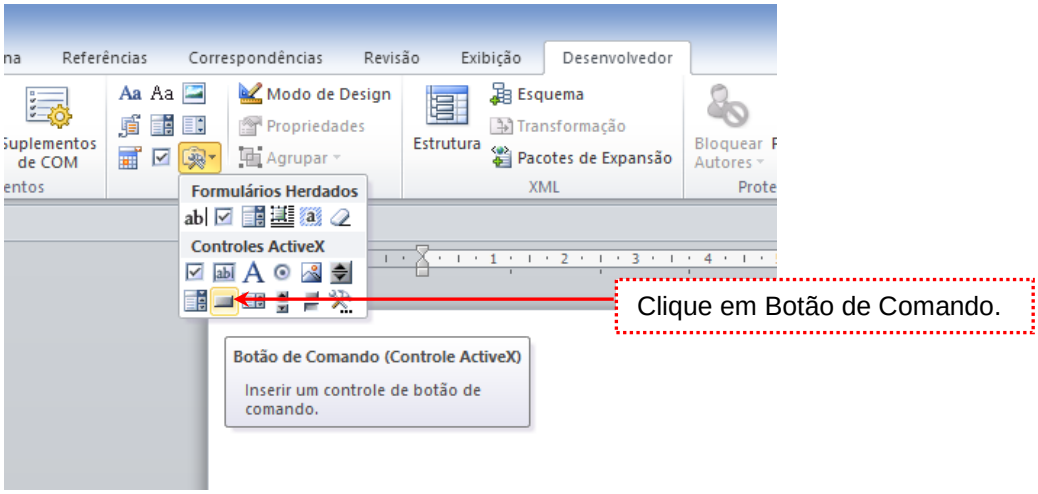
Mostraremos aqui como fazer as três opções.

1ª Opção: usando Controle ActiveX

Acione a Guia **Desenvolvedor** e no Grupo **Controles** clique em **Ferramentas Herdadas**.



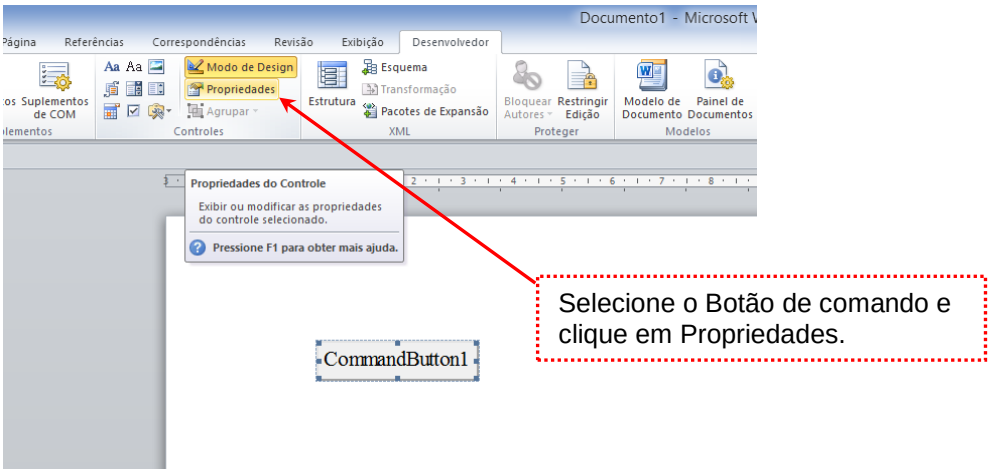
Em Controles ActiveX clique em Botão de Comando.

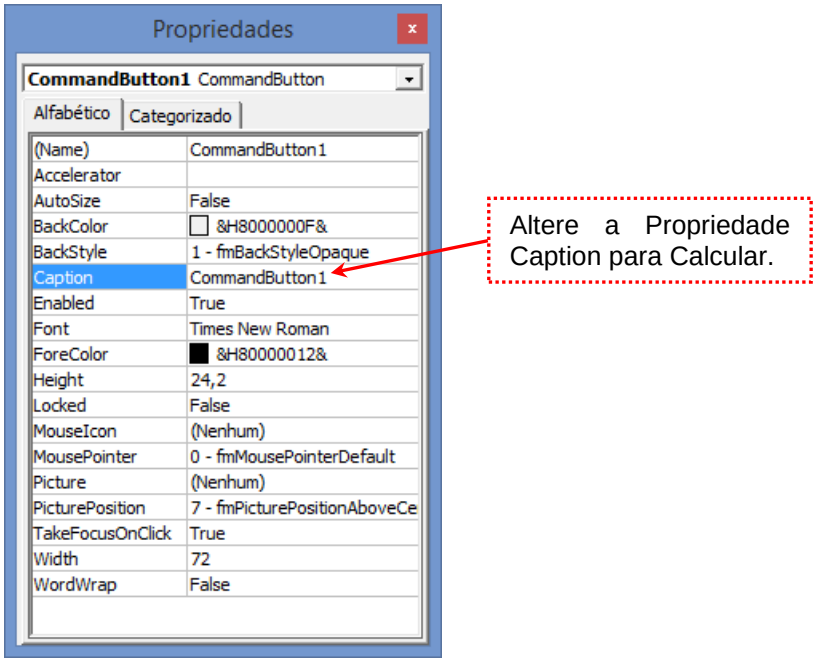


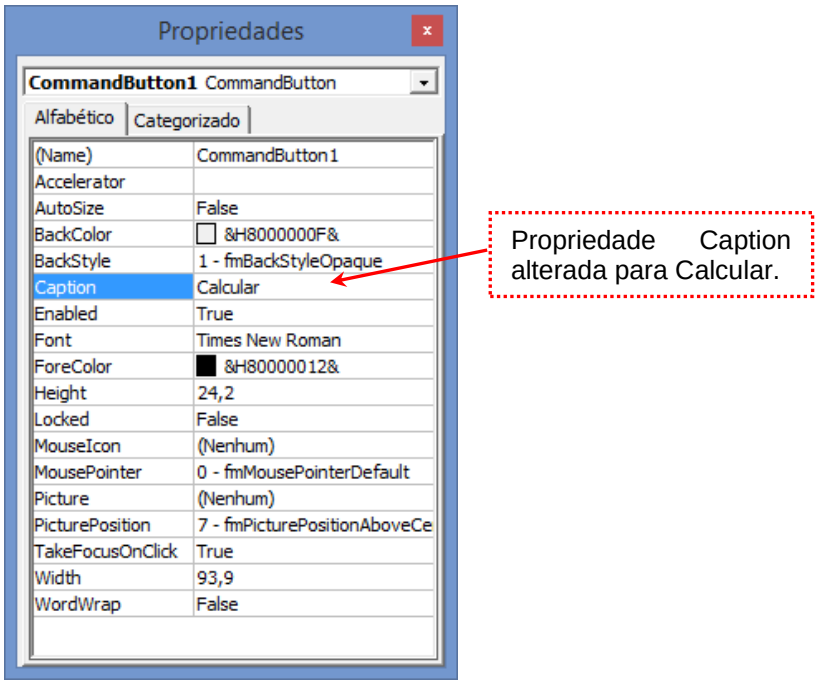
Por padrão o Botão de Comando possui sua **Propriedade Caption** como **CommandButton1** e caso seja inserido um novo botão ele será CommandButton2 e assim sucessivamente. Vamos alterar essa propriedade para "**Calcular**".



Selecione o Botão de Comando que acabamos de inserir no documento e no Grupo Controles clique em **Propriedades**.

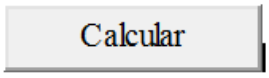






Você ainda pode, se quiser, alterar outras propriedades do botão como a sua cor através do propriedade BackColor; a fonte, estilo da fonte e tamanho através da propriedade Font, a cor da fonte através da propriedade ForeColor, etc.

Veja como ficou o nosso botão:



Agora é necessário colocar o código por trás do botão que acionará a macro **calculavalores**.

Clique duas vezes sobre o botão para abrir o Editor do Visual Basic ou selecione o botão e clique com o botão direito do mouse e selecione **Exibir Código.**

