

Índice

MACROS	5
O QUE É UMA MACRO?	6
CRIAR MACROS	6
<i>Gravar uma Macro</i>	6
Exercício	8
Procedimento BackGround do Excel	9
<i>Programação em Visual Basic for Applications</i>	10
EXECUTAR UMA MACRO	12
<i>Tecla de Atalho – Shortcut Key</i>	12
<i>Botão na Toolbar</i>	13
Associar uma Macro a um Botão	13
Dissociar uma Macro de um Botão	15
Run	16
<i>Comando no Menu</i>	17
Associação de uma Macro a um Comando do Menu	17
Dissociação	19
<i>Editor de Visual Basic for Applications</i>	20
REMOVER MACROS	21
<i>Remoção de Macros em Ambiente Excel</i>	21
<i>Remoção de Macros no Editor de VBA</i>	21
EDITOR DE VISUAL BASIC FOR APPLICATIONS	22
PROJECT EXPLORER	24
PROPERTIES WINDOW	26
JANELA DE EDIÇÃO	27
OBJECT BROWSER	28
HELP ON-LINE	28
AS CORES DO VBA	29
FUNÇÕES E SUBROTINAS	30
SUBROTINAS	32
<i>Definição de SubRotinas</i>	32
<i>Trabalhar no Editor de VBA – Criar uma SubRotina</i>	33
<i>Execução de uma SubRotina</i>	34
FUNÇÕES	34
<i>Definição de Funções</i>	34
<i>Definição do tipo de parâmetros e do tipo da função</i>	35
<i>Trabalhar no Editor de VBA – Criar uma Função</i>	36
<i>Execução de uma Função</i>	37
Execução dentro de uma Célula	38
Execução dentro de uma Rotina	40
DIFERENÇAS ENTRE FUNÇÕES E ROTINAS	42
REGRAS PARA A PASSAGEM DE PARÂMETROS	42
VARIÁVEIS	43

L

Excel – Macros e Visual Basic for Applications

MANUSEAMENTO COM VARIÁVEIS	43
<i>O que são variáveis?</i>	44
<i>Assignação de valores a variáveis:</i>	44
<i>Utilização de variáveis como se fossem valores:</i>	45
TIPOS DE VARIÁVEIS.....	45
DECLARAÇÃO DE VARIÁVEIS	47
VARIÁVEIS – VANTAGENS DA UTILIZAÇÃO	47
VARIÁVEIS DO TIPO OBJECTO	48
<i>Declaração da Variável Objecto</i>	48
<i>Atribuição de uma variável Objecto</i>	48
<i>Utilização Genérica da Variável Objecto</i>	49
<i>Utilização Específica da Variável Objecto</i>	49
VARIÁVEIS – DECLARAÇÃO OPCIONAL E O TIPO VARIANT	51
<i>Os Prós e Contras da Utilização do tipo Variants</i>	51
Prós.....	51
Contras	51
<i>Variáveis – Declaração Forçada</i>	52
VARIÁVEIS – TIPOS DEFINIDOS PELO UTILIZADOR - ESTRUTURAS	53
<i>Definição do Tipo</i>	53
<i>Utilização das Estruturas de Dados</i>	54
VARIÁVEIS – ARRAYS	55
<i>O que é um Array ?</i>	55
Array Uni-Dimensional.....	55
Array Bi-Dimensional	56
<i>Declaração de um array</i>	57
Arrays Uni-dimensionais	57
Arrays Bi-dimensionais.....	57
<i>Utilização de um Array</i>	58
Para aceder ao elemento.....	58
Atribuição de valores	58
<i>Option Base e Array Bounds</i>	59
CONSTANTES.....	ERRO! INDICADOR NÃO DEFINIDO.
O QUE SÃO CONSTANTES ?	ERRO! INDICADOR NÃO DEFINIDO.
INPUTBOX E MSGBOX.....	ERRO! INDICADOR NÃO DEFINIDO.
O QUE SÃO ?	ERRO! INDICADOR NÃO DEFINIDO.
INPUTBOX.....	ERRO! INDICADOR NÃO DEFINIDO.
<i>O que faz...</i>	<i>Erro! Indicador não definido.</i>
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Parâmetros</i>	<i>Erro! Indicador não definido.</i>
MSGBOX.....	ERRO! INDICADOR NÃO DEFINIDO.
<i>O que faz...</i>	<i>Erro! Indicador não definido.</i>
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Parâmetros</i>	<i>Erro! Indicador não definido.</i>
<i>Valores Produzidos</i>	<i>Erro! Indicador não definido.</i>
DOMÍNIO DAS VARIÁVEIS, CONSTANTES E ROTINAS ..	ERRO! INDICADOR NÃO DEFINIDO.
O QUE É O DOMÍNIO?	ERRO! INDICADOR NÃO DEFINIDO.
DOMÍNIO DAS VARIÁVEIS.....	ERRO! INDICADOR NÃO DEFINIDO.
<i>Âmbito do Procedimento</i>	<i>Erro! Indicador não definido.</i>
<i>Âmbito do Módulo</i>	<i>Erro! Indicador não definido.</i>
<i>Âmbito do Projecto</i>	<i>Erro! Indicador não definido.</i>
DOMÍNIO DAS CONSTANTES	ERRO! INDICADOR NÃO DEFINIDO.
<i>Âmbito do Procedimento</i>	<i>Erro! Indicador não definido.</i>
<i>Âmbito do Módulo</i>	<i>Erro! Indicador não definido.</i>
<i>Âmbito do Projecto</i>	<i>Erro! Indicador não definido.</i>

L

Excel – Macros e Visual Basic for Applications

DOMÍNIO DE SUBROTINAS E FUNÇÕES	ERRO! INDICADOR NÃO DEFINIDO.
ESTRUTURAS DE CONTROLO	ERRO! INDICADOR NÃO DEFINIDO.
O QUE SÃO ESTRUTURAS DE CONTROLO?	ERRO! INDICADOR NÃO DEFINIDO.
QUAIS AS ESTRUTURAS... ..	ERRO! INDICADOR NÃO DEFINIDO.
IF-THEN-ELSE	ERRO! INDICADOR NÃO DEFINIDO.
<i>Função IF do Excel</i>	<i>Erro! Indicador não definido.</i>
<i>Sintaxe da Estrutura If-Then-Else</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicação Prática</i>	<i>Erro! Indicador não definido.</i>
<i>A instrução adicional ElseIf</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicação Prática</i>	<i>Erro! Indicador não definido.</i>
FOR – NEXT	ERRO! INDICADOR NÃO DEFINIDO.
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicação Prática</i>	<i>Erro! Indicador não definido.</i>
<i>A Função das Variáveis:</i>	<i>Erro! Indicador não definido.</i>
<i>Construção do Ciclo:</i>	<i>Erro! Indicador não definido.</i>
<i>Tradução Integral</i>	<i>Erro! Indicador não definido.</i>
<i>Funcionamento do Ciclo:</i>	<i>Erro! Indicador não definido.</i>
<i>Perigos associados à utilização do ciclo For-Next:</i>	<i>Erro! Indicador não definido.</i>
<i>Outra Aplicação</i>	<i>Erro! Indicador não definido.</i>
WHILE-WEND	ERRO! INDICADOR NÃO DEFINIDO.
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicação Prática</i>	<i>Erro! Indicador não definido.</i>
<i>A Função das Variáveis:</i>	<i>Erro! Indicador não definido.</i>
<i>Construção do Ciclo:</i>	<i>Erro! Indicador não definido.</i>
<i>Tradução Integral</i>	<i>Erro! Indicador não definido.</i>
<i>Funcionamento do Ciclo</i>	<i>Erro! Indicador não definido.</i>
<i>Perigos associados à utilização do ciclo While-Wend</i>	<i>Erro! Indicador não definido.</i>
<i>Outra Aplicação</i>	<i>Erro! Indicador não definido.</i>
DO – LOOP	ERRO! INDICADOR NÃO DEFINIDO.
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicações Práticas</i>	<i>Erro! Indicador não definido.</i>
SELECT CASE	ERRO! INDICADOR NÃO DEFINIDO.
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicação Prática</i>	<i>Erro! Indicador não definido.</i>
<i>Construção da Estrutura</i>	<i>Erro! Indicador não definido.</i>
FOR – EACH – NEXT	ERRO! INDICADOR NÃO DEFINIDO.
<i>Sintaxe</i>	<i>Erro! Indicador não definido.</i>
<i>Aplicações Práticas</i>	<i>Erro! Indicador não definido.</i>
<i>Utilizando Arrays</i>	<i>Erro! Indicador não definido.</i>
<i>Construção do Ciclo</i>	<i>Erro! Indicador não definido.</i>
<i>Utilizando Coleções de Objectos</i>	<i>Erro! Indicador não definido.</i>
COLECCÕES DE OBJECTOS E OBJECTOS	ERRO! INDICADOR NÃO DEFINIDO.
O QUE SÃO OBJECTOS ?	ERRO! INDICADOR NÃO DEFINIDO.
OBJECTOS: PROPRIEDADES, MÉTODOS E EVENTOS	ERRO! INDICADOR NÃO DEFINIDO.
<i>Propriedades</i>	<i>Erro! Indicador não definido.</i>
<i>Métodos</i>	<i>Erro! Indicador não definido.</i>
<i>Eventos</i>	<i>Erro! Indicador não definido.</i>
OBJECTOS MAIS UTILIZADOS NO EXCEL	ERRO! INDICADOR NÃO DEFINIDO.
Propriedades	Erro! Indicador não definido.
Métodos	Erro! Indicador não definido.
Propriedades	Erro! Indicador não definido.
Métodos	Erro! Indicador não definido.
Propriedades	Erro! Indicador não definido.
Métodos	Erro! Indicador não definido.
Propriedades	Erro! Indicador não definido.
Métodos	Erro! Indicador não definido.

Excel – Macros e Visual Basic for Applications

OBJECTOS SINGULARES VS	COLECÇÕES DE OBJECTOS.....	ERRO! INDICADOR NÃO DEFINIDO.
INDEXAÇÃO DE COLECÇÕES POR NÚMERO OU NOME		ERRO! INDICADOR NÃO DEFINIDO.
<i>Indexação com Base em Números.....</i>		<i>Erro! Indicador não definido.</i>
<i>Indexação com Base no Nome.....</i>		<i>Erro! Indicador não definido.</i>
Vantagem		Erro! Indicador não definido.
<i>O Objecto Range – uma excepção.....</i>		<i>Erro! Indicador não definido.</i>
Tratamento como objecto:		Erro! Indicador não definido.
Tratamento como colecção de objectos:		Erro! Indicador não definido.
REFERÊNCIA IMPLÍCITA.....		ERRO! INDICADOR NÃO DEFINIDO.
<i>Declaração implícita da aplicação:</i>		<i>Erro! Indicador não definido.</i>
<i>Declaração implícita do Workbook:</i>		<i>Erro! Indicador não definido.</i>
<i>Declaração implícita da Worksheet:.....</i>		<i>Erro! Indicador não definido.</i>
<i>Nível de referência a privilegiar</i>		<i>Erro! Indicador não definido.</i>
MISCELLANEOUS.....		ERRO! INDICADOR NÃO DEFINIDO.
A INSTRUÇÃO WITH.....		ERRO! INDICADOR NÃO DEFINIDO.
<i>Aplicação Prática.....</i>		<i>Erro! Indicador não definido.</i>
OUTRAS FUNÇÕES ÚTEIS DO VBA		ERRO! INDICADOR NÃO DEFINIDO.

Macros

O QUE É UMA MACRO?

Uma macro é um pequeno programa que contém uma lista de instruções a realizar no Excel. Como sendo um repositório de operações, uma macro pode executar um conjunto de tarefas através de um único procedimento o qual pode ser invocado rapidamente.

As instruções que formam o corpo da macro são escritas num código próprio para que o computador as possa entender, essa linguagem é designada por VBA – Visual Basic for Applications.

CRIAR MACROS

Existem duas possibilidades de criação de macros:

- Através do Gravador de Macros
- Utilizando o editor e programando em Visual Basic for Applications

Gravar uma Macro

1. Tools / Macro / Record New Macro. O Excel exibirá a caixa de diálogo da fig.1.

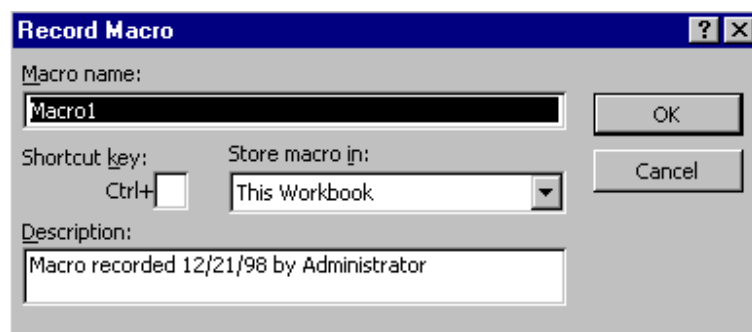


Fig.1 – caixa de diálogo para a gravação de macros.

Excel – Macros e Visual Basic for Applications

2. O nome da Macro será constituído por um conjunto de caracteres que identificarão a Macro e a sua funcionalidade.
3. Shortcut Key – é composto por uma combinação de teclas que poderão ser utilizadas para executar uma macro.
4. Opte por armazenar a macro numa das seguintes alternativas: This Workbook, New Workbook ou Personal Workbook, sendo que cada uma corresponde a macros de âmbito diferente. Assim deverão armazenar a macro na opção ThisWorkBook para que ela esteja activa sempre que o documento estiver aberto, e para que ela esteja armazenada no próprio ficheiro não correndo o risco de ficar armazenada nas Macros do Excel. (na rede do ISCTE é impossível gravar a macro no Personal WorkBook.)
5. No campo Description, introduza um comentário à função – este poderá ser importante para que não seja esquecido o respectivo objectivo e funcionalidade.
6. Clique em OK para iniciar a gravação da macro – neste momento será exibida uma toolbar semelhante à da figura 2, e o ícone do rato será transformado numa cassete, indicando o estado de gravação.



Fig.2 – Toolbar exibida para a gravação de macros.

7. Na toolbar Stop Record, existem dois botões: Stop Recording e Relative Reference – O botão de Stop Recording termina a gravação da macro, o botão de Relative Reference selecciona o modo de gravação da macro – se é feito com base em referências relativas (botão seleccionado) ou referências absolutas.
8. No caso da toolbar Stop Record desaparecer, poderá voltar a exibi-la fazendo no menu a sequência: View / Toolbars / Stop Record (selecione a toolbar). Caso a toolbar não apareça listada a gravação de macros não está activa.

Nota: Tome atenção aos passos que dá quando está a gravar pois tudo será registado, o que significa que quando for executar a macro, esses procedimentos serão efectuados.

EXERCÍCIO

Objectivo: Gravar uma macro testando a diferença entre a execução de macros com referências absolutas e relativas.

1ª Fase: Macro com referências Absolutas

1. Accione a gravação da macro. Atribua-lhe o Short Key Ctrl+P
2. Certifique-se que o botão Relative Reference está desactivado.
3. Clique na célula B3.
4. Escreva *ISCTE*
5. Formate a célula para Bold, Itálico, tamanho 18, Small Caps,... (utilize o Format / Font)
6. Na célula B4 escreva: *Av. Forças Armadas*
7. Na célula B5 escreva: *1700 Lisboa*
8. Pare a gravação da macro. – A macro está criada.
9. Apague tudo o que escreveu nas células da coluna B.
10. Clique na célula L8.
11. Carregue simultaneamente nas teclas Ctrl e P
12. O que aconteceu ?

2ª Fase: Macro com referências Relativas

1. Seleccione a célula D5.
2. Accione a gravação da macro. Atribua-lhe o Short Key Ctrl+R
3. Seleccione o botão Relative Reference.
4. Escreva *ISCTE* na célula que havia sido seleccionada.
5. Formate a célula para Bold, Itálico, tamanho 18, Small Caps,... (utilize o Format / Font)
6. Na célula D6 escreva: *Av. Forças Armadas*
7. Na célula D7 escreva: *1700 Lisboa*
8. Pare a gravação da macro. – A macro está criada.
9. Apague tudo o que escreveu nas células da coluna D.
10. Clique na célula L8.
11. Carregue simultaneamente nas teclas Ctrl e R
12. O que aconteceu ? Porquê ?

PROCEDIMENTO BACKGROUND DO EXCEL

O Excel quando se grava uma macro cria um objecto designado por *module* no workbook onde regista todas as operações gravadas em linguagem ***Visual Basic for Applications - VBA***. Este module não aparece no Excel com as restantes Sheets.

Para ser visualizado é necessário abrir o **Editor de Visual Basic for Applications**:

1. Tools / Macro / Macros
2. Selecciona-se a Macro e Clica-se no botão Edit
3. Poderá visualizar na área do lado direito o código VBA que está subjacente às macros que entretanto gravou. Aí poderá executar as mesmas tarefas que num editor de texto normal: escrever, apagar, copiar, mover,... mas instruções de VBA.
4. Tente fazer a leitura do que está escrito e compreenda o procedimento da macro.
5. Para regressar ao Excel basta File / Close and return to Microsoft Excel.

Excel – Macros e Visual Basic for Applications

Programação em Visual Basic for Applications

1. Para aceder ao editor de Visual Basic for Applications: Tools / Macro / Visual Basic Editor (para se familiarizar mais com o editor consulte o capítulo *Editor Visual Basic for Applications* e o capítulo *Funções e SubRotinas*)

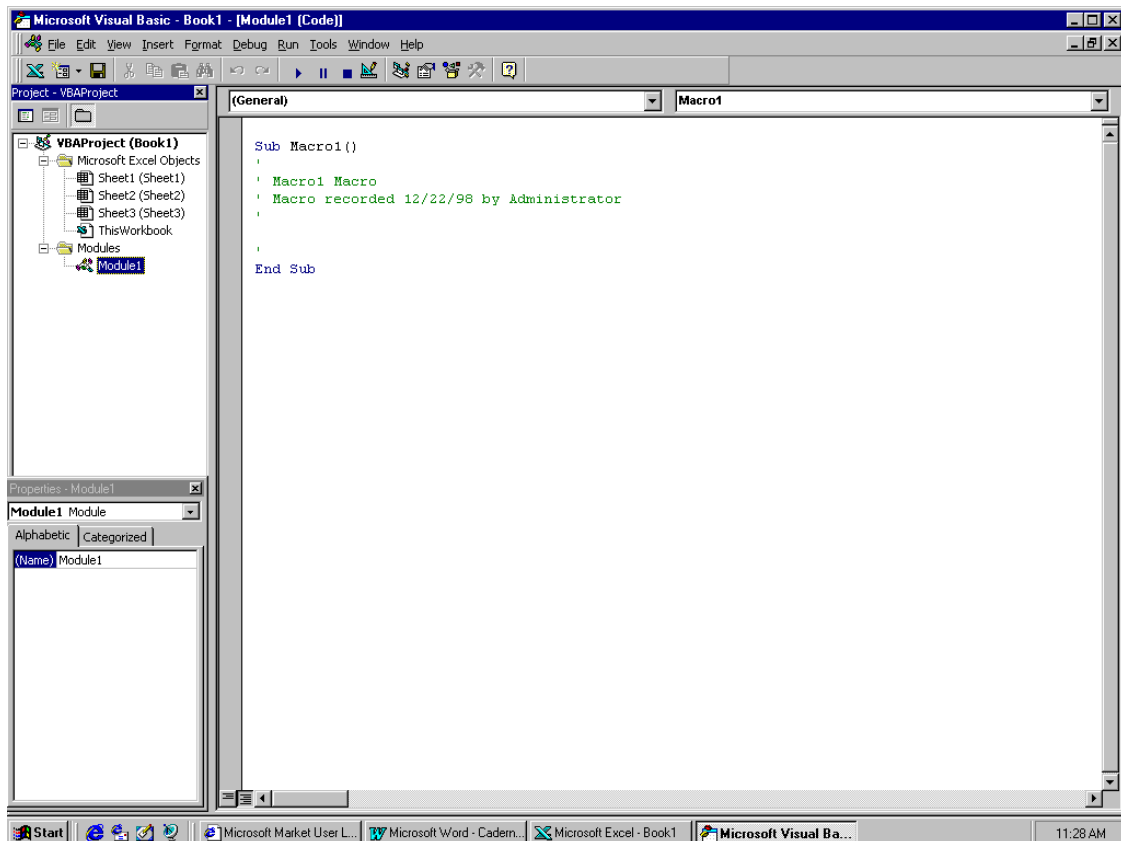


Figura 3 – Editor de Visual Basic for Applications

2. Para inserir um módulo faça Insert / Module – isto porque as macros que construir deverão ser escritas dentro de um módulo (repare na Figura 3 que o Module 1 está seleccionado na janela de projecto e do lado direito tem-se uma área de edição onde poder-se-ão escrever as macros a executar)
3. Pode agora programar os procedimentos manualmente ou com recurso a algumas funcionalidades do Editor:

Excel – Macros e Visual Basic for Applications

- a) Insert / Procedure – insere um procedimento para o qual deverá indicar o tipo de procedimento.
 - b) Quando estiver a introduzir as variáveis haverá de ser exibida a lista de tipos de variáveis possível.
4. Sempre que precisar de construir uma macro mais ou menos complicada, se não conhecer muito de VBA poderá começar por gravar no EXCEL pequenas macros com operações muito simples e depois no editor de VBA tentar compreender a sua funcionalidade e assim construir uma macro cada vez mais complexa, completa e adequada às suas exigências.
5. Poderá em cada macro fazer uma chamada a uma outra macro, bastando para tal escrever o respectivo nome.
6. No Editor de Visual Basic for Applications poderá encontrar ajuda para o desenvolvimento do seu procedimento. Assim:
- 6.1. Clique no Ícone do Object Browser ou View/Object Browser ou F2
 - 6.2. Na Caixa de Drop-Down onde aparece referido <All Libraries>, seleccione a aplicação para a qual pretende ajuda – neste caso Excel. Convém referir que poderá utilizar as funções de toda e qualquer aplicação.
 - 6.3. Na área intitulada por Classes aparecem todos os objectos específicos para o manuseamento da aplicação seleccionada. A classe especial designada por Globals refere-se às funções que estão disponíveis na aplicação independentemente dos objectos seleccionados.
 - 6.4. Seleccione um dos objectos e visualize do lado direito os Members of “<Elemento seleccionado>”
 - 6.5. Para cada membro da classe dispõe de um help on-line que o esclarece sobre a respectiva função e funcionamento, dando exemplo que poderá testar. Para tal basta clicar sobre o botão de Help da janela do Object Browser.

EXECUTAR UMA MACRO

A execução de uma macro pode ser concretizada de diversas formas:

- Tecla de Atalho – Shortcut Key
- Botão na Toolbar
- Run
- Comando no Menu
- Editor de Visual Basic for Applications

Tecla de Atalho – Shortcut Key

A associação a teclas de atalho é realizada aquando da criação da macro.

Botão na Toolbar

ASSOCIAR UMA MACRO A UM BOTÃO

1. View / Toolbars / Customize
2. Na janela do Customize seleccione o Commands Tab
3. Seleccione a categoria Macro (visualizará a caixa de diálogo exibida na Figura 4)

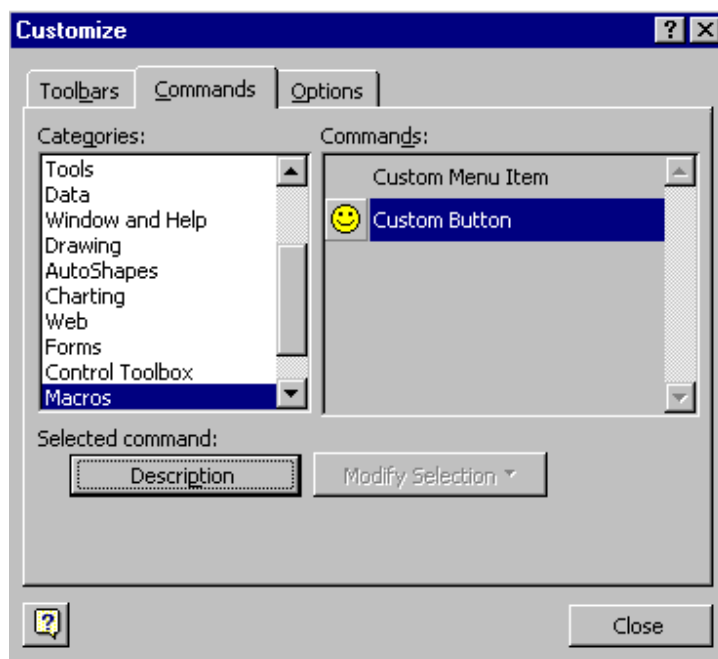


Fig.4 – caixa de diálogo para assignação de botão a macros

Excel – Macros e Visual Basic for Applications

4. Na área respeitante aos Commands serão exibidos dois itens: Custom Menu Item e Custom Button. Seleccione este segundo e arraste-o com o rato até à Toolbar onde o pretende inserir. Atenção só poderá inserir o botão quando o ponteiro do rato ficar com a forma **I**. Nesse momento libertará o botão do rato utilizado para o arrastamento e verificará que será criado um novo botão.
5. Na janela de Customize poderá ainda utilizar dois botões que se encontram na área do Selected Command:
 - a) Description – que exhibe um texto explicando o que o comando seleccionado faz.
 - b) Modify Selection – semelhante ao clique sobre o botão criado, exhibe uma série de tarefas possíveis para configuração do botão (ver operações seguintes).

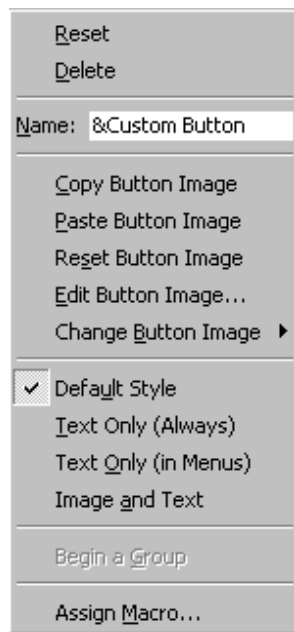


Fig.5 – Menu para configuração do botão da toolbar

Excel – Macros e Visual Basic for Applications

6. Clique sobre o botão Modify Selection- abrindo um menu de tarefas possíveis para a configuração do botão – Ver Figura 5
- a) No último agrupamento de configurações possíveis existe uma opção designada por Assign Macro. Esta opção permite indicar qual a macro que deverá ser executada sempre que se clica no botão.
 - b) No terceiro agrupamento existem 4 estilos diferentes de exibir o botão: só com texto, com texto e imagem ou somente com imagem. Se seleccionar o estilo Texto e Imagem, será exibido no botão para além da imagem o nome associado ao botão.
 - c) Na opção Name indique o nome que pretende ver associado ao botão, por defeito aparece o nome da macro. (repare que aparece um & atrás da letra que aparece a sublinhado)
 - d) Para alterar a imagem associada ao botão poderá: optar por uma imagem diferente, alterar a que está a visualizar ou a construir a sua. Para isso clique sobre a tarefa de Change Button Image, seleccione o logotipo pretendido, se nenhum o satisfizer e pretender criar o seu seleccione o logotipo que representa um quadrado vazio. Para o poder (re)desenhar a imagem, na lista de tarefas disponibilizada pelo botão Modify Selection opte por Edit Button Image e crie a sua imagem que poderá reeditar.

DISSOCIAR UMA MACRO DE UM BOTÃO

1. Tools / Customize
2. Arraste o botão da toolbar até ao documento
3. Solte-o

Excel – Macros e Visual Basic for Applications

Run

1. Tools / Macro / Macros
2. Na caixa de diálogo Macros selecciona-se a macro pretendida na lista da Macro Name (Figura 6)
3. Clique sobre o botão Run

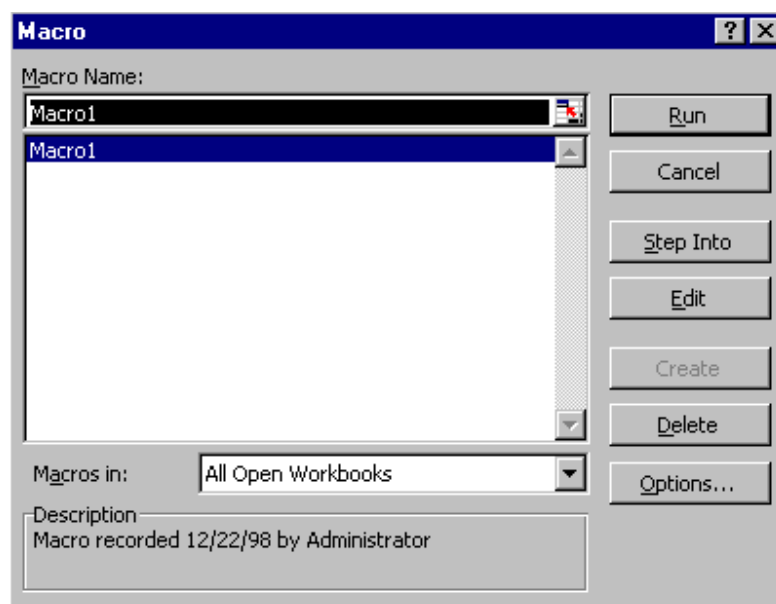


Fig.6– Janela para selecção da macro a executar

Comando no Menu

ASSOCIAÇÃO DE UMA MACRO A UM COMANDO DO MENU

1. View / Toolbars / Customize
2. Na janela do Customize encontra-se no Commands Tab
3. Seleccione a categoria Macro
4. Na área respeitante aos Commands será exibido um item de Custom Menu Item, seleccione-o e arraste-o com o rato até a uma posição do menu que lhe interesse – por exemplo pode introduzir numa das listas de opções do menu (File, View,...) ou então criar como uma nova opção do Menu..

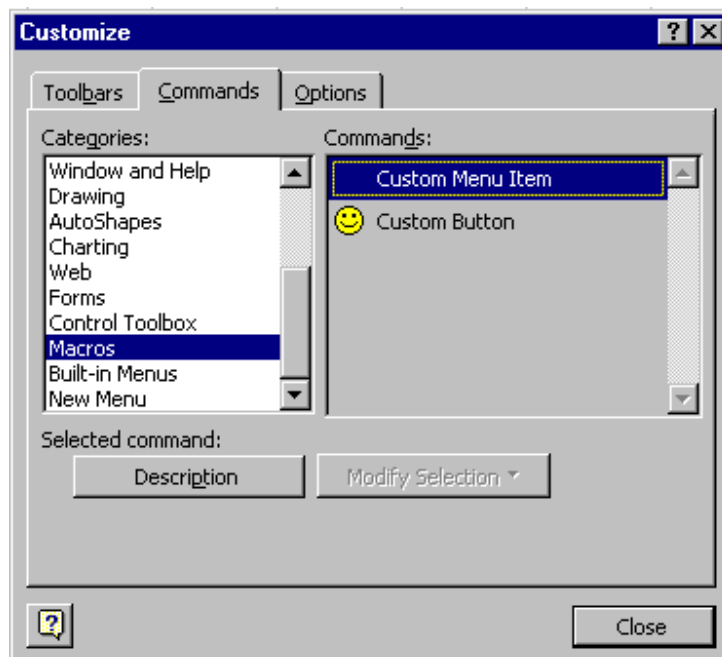


Fig.7 – Caixa de diálogo para atribuir uma macro a um comando do menu

5. Se pretender criar uma nova lista no menu deverá :

Excel – Macros e Visual Basic for Applications

- a) Seleccionar a categoria New Menu
- b) Na área dos Commands será exibida a opção New Menu, que deverá arrastar até à barra dos menus
- c) Poderá alterar o seu nome clicando no botão de Modify Selection
- d) Esta nova lista terá o mesmo comportamento que a outras

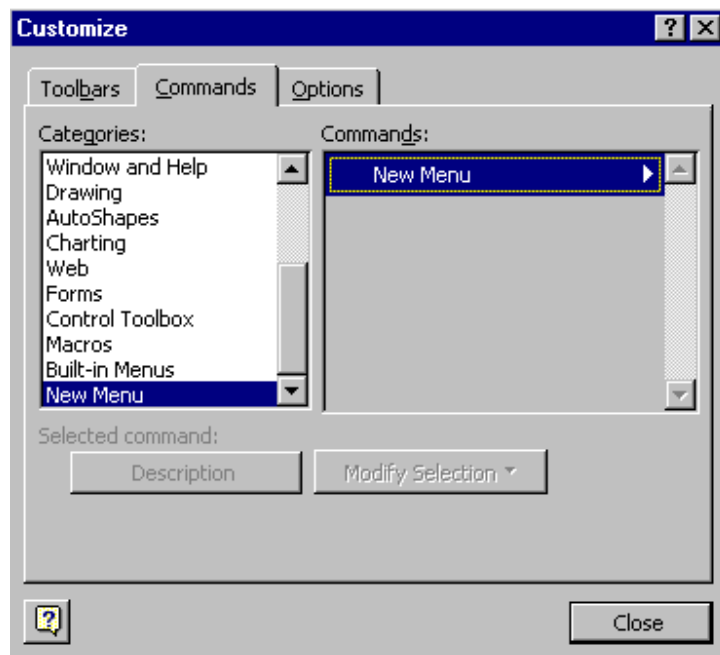


Fig.8 – Caixa de diálogo para criar um novo menu na barra dos menus

- 6. Na janela de Customize poderá ainda utilizar dois botões que se encontram na área do Selected Command:
 - c) Description – que exhibe um texto explicando o que o comando seleccionado faz.
 - d) Modify Selection – semelhante ao clique sobre o menu ou comando de menu criado, exhibe uma série de tarefas possíveis para configuração (Ver o item 6 do capítulo *Associar uma Macro a um Botão*)

DISSOCIAÇÃO

1. Tools / Customize
2. Arraste o Menu ou Comando do Menu até ao documento e solte-o

Editor de Visual Basic for Applications

1. Tools / Macro / Visual Basic Editor - para aceder ao Editor
2. Posicionando-se no corpo de uma macro, na janela do lado direito inferior, poderá executar a macro através de: Run / Run Sub-UserForms (figura 9) ou botão Run.

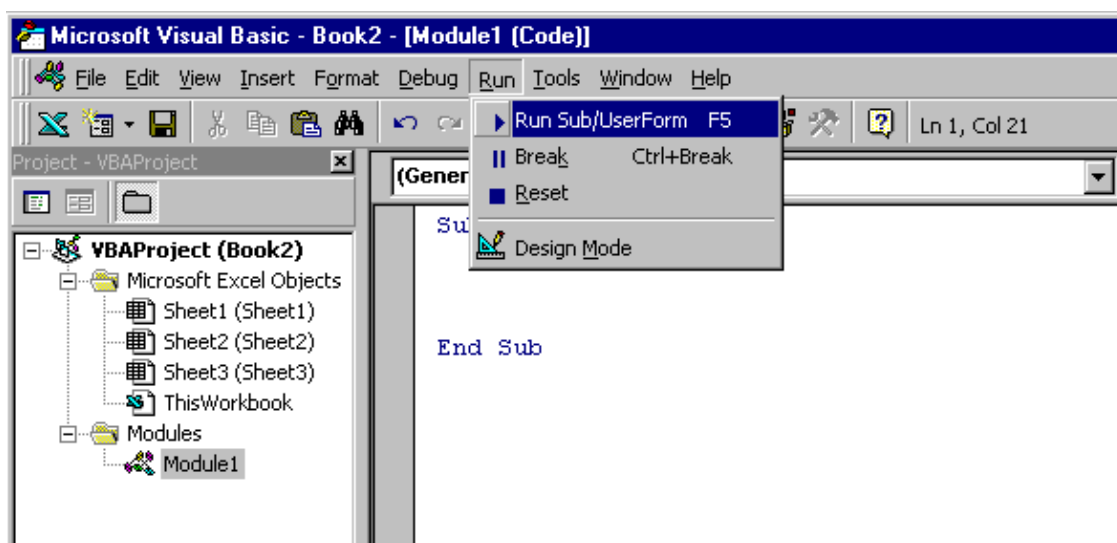


Fig.9 – Execução de uma macro no Editor de Visual Basic for Applications

REMOVER MACROS

A remoção das macros poderá ser feita:

- no ambiente Excel, ou
- no Editor de VBA

Remoção de Macros em Ambiente Excel

1. Tools / Macro / Macros
2. Selecciona-se a Macro a remover
3. Clica-se no botão Delete
4. Pede a confirmação e apaga se confirmar

Remoção de Macros no Editor de VBA

2. Tools / Macro / Visual Basic Editor - para aceder ao Editor
3. Podem-se apagar as macros que se encontrem na janela do lado direito inferior e que têm início com a palavra Sub e fim com as palavras End Sub (Ver capítulo de funções e procedimentos)

Editor de Visual Basic for Applications

Excel – Macros e Visual Basic for Applications

Para aceder ao editor de Visual Basic for Applications: Tools / Macro / Visual Basic Editor

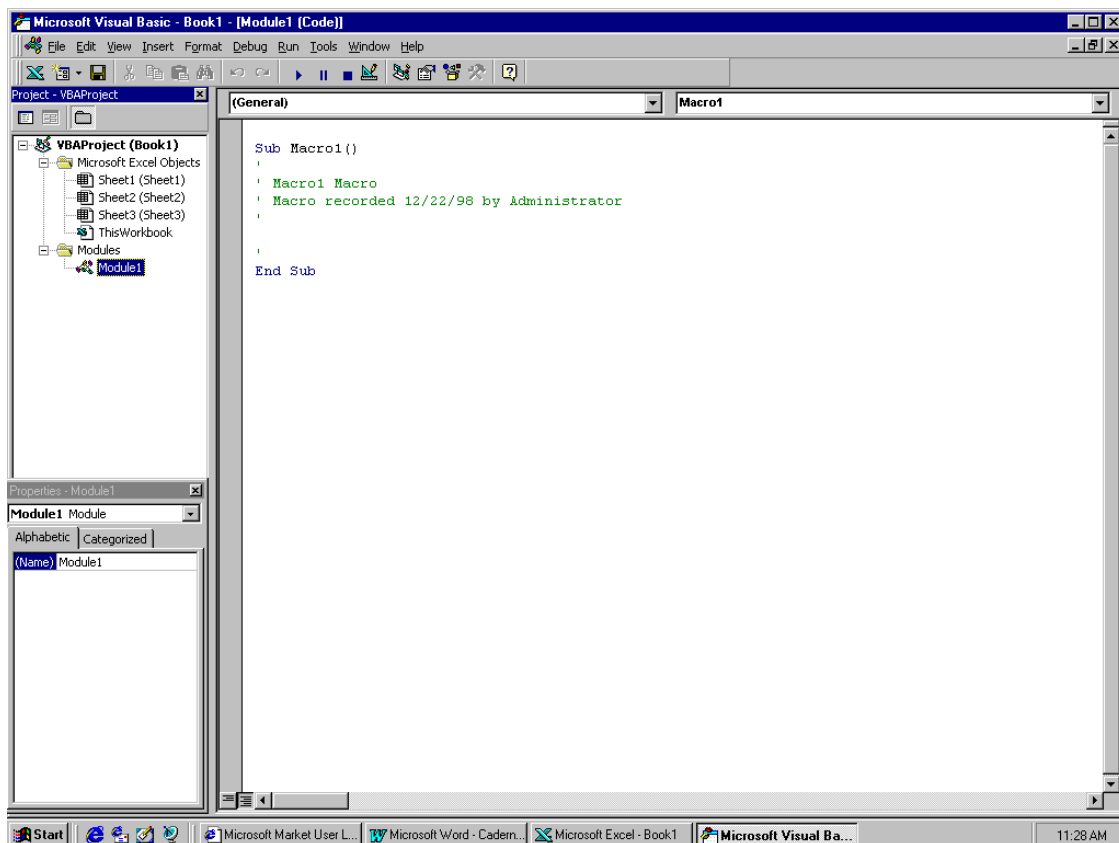


Figura 10 – Editor de Visual Basic for Applications

Encontrará o écran dividido em três grandes áreas: Project Explorer, Properties Window e do lado direito a janela de edição de texto.

PROJECT EXPLORER

Se a janela não estiver visível active-a através do View / Project Explorer ou utilizando o botão ou combinações de letras associadas a esta tarefa.

Nesta janela poderá visualizar a hierarquia dos projectos de Visual Basic for Applications activos nesse momento.

Entre eles deverá encontrar um cujo nome corresponde ao nome do WorkBook do Excel com que está a trabalhar. E.g. VBAProject (Book1). É dentro deste projecto que deverá trabalhar para que todas as funcionalidades que implemente estejam nele activas sempre que necessário.

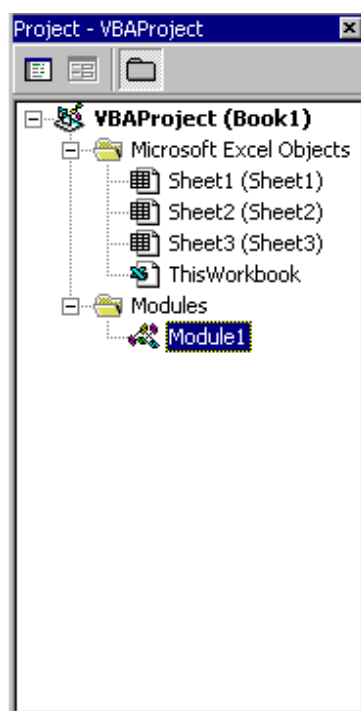


Figura 11 – Janela do Project Explorer

Excel – Macros e Visual Basic for Applications

Assim se fizer o desdobramento do seu VBAProject, encontrará um folder cuja designação é Microsoft Excel Objects, e um outro designado Modules (se este não aparecer significa que o seu projecto ainda não possui qualquer macro implementada. Para criar este folder deverá fazer: Insert / Module).

No folder do Microsoft Excel Objects, encontrará todos os objectos que fazem parte do seu documento: as WorkSheets e o WorkBook (que no fundo é o conjunto de WorkSheets). Se clicar duplamente em cada um destes objectos, uma nova janela será visualizada na área da direita, em cada uma dessas áreas poderá definir a acção a ser executada em função dos eventos de cada objecto.

No Folder Modules, aparecerá o conjunto de ficheiros (módulos) onde poderá programar as suas macros. Clicando duplamente em cada um dos módulos indicados poderá visualizar as macros, que o compõem, na janela da direita.

PROPERTIES WINDOW

Se a janela das propriedades não estiver visível active-a através do View / Properties Window ou utilizando o botão ou combinações de letras associadas a esta tarefa.

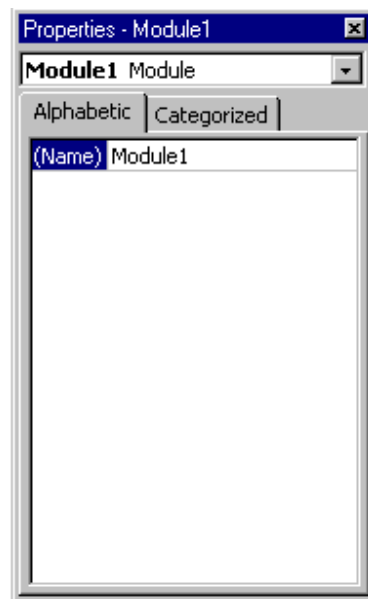


Figura 12 – Janela de propriedades

Nesta janela poderá visualizar e alterar as propriedades que definem cada objecto: o nome - neste caso.

JANELA DE EDIÇÃO

A janela de edição exibirá a cada momento o código em Visual Basic for Applications associado ao elemento seleccionado na janela do Project Explorer.

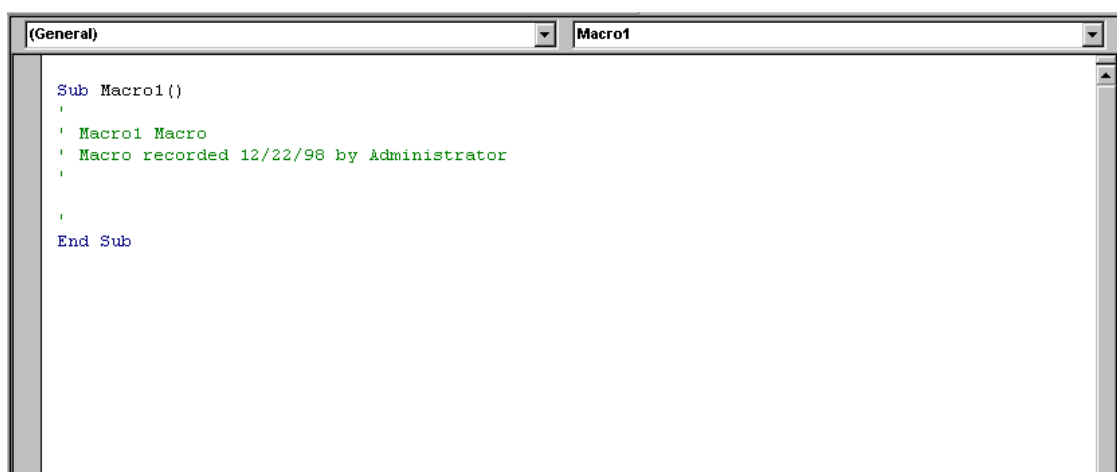


Figura 13 – Janela de edição

OBJECT BROWSER

No Editor de Visual Basic for Applications poderá encontrar ajuda para o desenvolvimento do seu procedimento. Assim:

- Clique no Ícone do Object Browser ou View/Object Browser ou F2
- Na Caixa de Drop-Down onde aparece referido <All Libraries>, seleccione a aplicação para a qual pretende ajuda – neste caso Excel. Convém referir que poderá utilizar as funções de toda e qualquer aplicação.
- Na área intitulada por Classes aparecem todos os objectos específicos para o manuseamento da aplicação seleccionada. A classe especial designada por Globals refere-se às funções que estão disponíveis na aplicação independentemente dos objectos seleccionados.
- Seleccione um dos objectos e visualize do lado direito os Members of “ <Elemento seleccionado>”
- Para cada membro da classe dispõe de um help on-line que o esclarece sobre a respectiva função e funcionamento, dando um exemplo que poderá testar. Para tal basta clicar sobre o botão de Help da janela do Object Browser.

HELP ON-LINE

O Editor de Visual Basic for Applications, disponibiliza ao utilizador um sistema de ajuda constante. Assim, quando se constroem procedimentos, na janela de edição, à medida que a linguagem é escrita o editor abre oportunamente listas de opções para ajudar a escrever o código.

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 rectificar 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	Comentário 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.

Funções e SubRotinas

Excel – Macros e Visual Basic for Applications

Como foi referido anteriormente, quando se grava uma macro no Excel, este tem um comportamento em background que realiza a respectiva codificação para Visual Basic for Applications. Este resultado é bem visível quando procedemos à edição de uma macro no Editor de Visual Basic for Applications. Cada macro que se cria tem um comportamento concreto e autónomo relativamente a outras macros implementadas, e tem como objectivo executar um determinado número de instruções¹ que respondam às exigências do utilizador.

Cada macro criada dá origem a um procedimento ou rotina. Existem dois tipos de rotinas:

- as subrotinas ou rotinas Sub, e
- as funções.

¹ Entenda-se instrução como uma tarefa a executar que corresponde a uma linha de código.

SUBROTINAS

Definição de SubRotinas

As SubRotinas são aquelas cuja definição é delimitada pelas palavras-chave *Sub* e *EndSub*. Assim se reparar todas as macros que grava no Excel são deste tipo. Repare ainda como é que são definidas:

Sub <nome_da_macro> ()

<corpo_da_macro>

End Sub

Estas SubRotinas são designadas pelo nome² que lhe atribuímos e não recebem parâmetros³ do exterior, têm como função desempenhar um conjunto de tarefas que compõem o seu corpo. O corpo da macro, é assim composto por um conjunto de instruções, sendo que cada instrução diferente necessita de estar numa linha diferente. Contudo, quando se trata de instruções demasiado grandes o editor faz a sua partição por diversas linhas, recorrendo ao operador “_”, por forma a facilitar a leitura.

² O nome da rotina pode ser qualquer um desde que não contenha espaços, comece por caracteres alfa

³ Para ter a noção do que são parâmetros recorde-se do funcionamento das funções do Excel, que para executarem determinada função necessitavam de receber parâmetros que colocamos entre parêntesis separados por vírgulas.

Trabalhar no Editor de VBA – Criar uma SubRotina

Para criar uma SubRotina é necessário que exista um módulo onde se possa escrever. Uma vez posicionado nesse módulo poderá:

- Escrever a macro integralmente, ou
- Recorrer ao Insert / Procedure para que o Visual Basic for Applications lhe crie a estrutura (Figura 14)

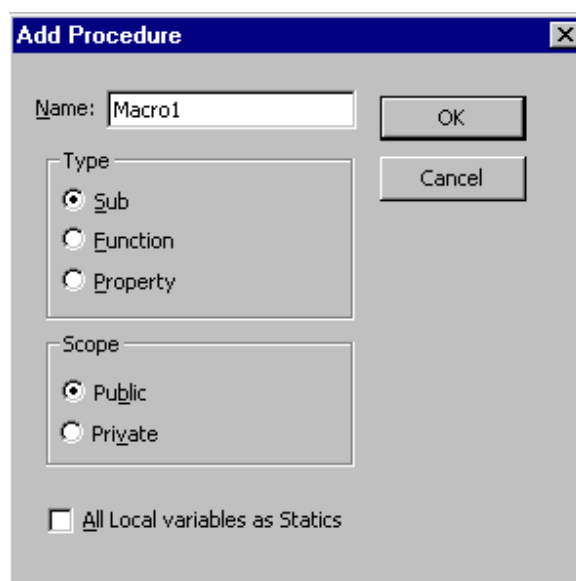


Figura 14 – Caixa de Diálogo para a criação de uma nova rotina

Assim deverá indicar o tipo de rotina a criar – Sub – e o nome que pretende dar à macro (ou rotina ou procedimento ou SubRotina)⁴. Automaticamente ele criará a estrutura da rotina, neste caso ficaria:

```
Sub Macro1( )
```

```
End Sub
```

⁴ Nomes alternativos para fazer referência a um conjunto de instruções.

Execução de uma SubRotina

SubRotina e Macro são duas designações para a mesma realidade, portanto tudo o que foi referido relativamente a Macros é válido também para as SubRotinas. (Consultar capítulo das Macros)

Neste contexto a execução de SubRotinas segue o mesmo mecanismo das macros. Porém neste momento já se pode referir uma nova forma de executar as macros ou SubRotinas – dentro de outras rotinas, i.e., quando se escreve dentro de uma rotina o nome de outra a execução da rotina tem continuidade na execução da rotina que está a ser invocada.

FUNÇÕES

Definição de Funções

Funções são rotinas cuja definição começa com a palavra-chave *Function* e termina com as palavras *End Function*. Todas as funções que utiliza no Excel são deste tipo de rotina. A sua definição tem a estrutura seguinte:

Function <Nome da Função> (<parametro1>, <parametro2>,...)

...

<Nome da Função> = <Valor / Expressão>

...

End Function

A função é identificada pelo nome, pelo número e tipo de parâmetros recebidos, e tem como objectivo executar um conjunto de instruções e produzir um valor final. Isto é, sempre que se pretender executar uma função é sabido à priori que ela produzirá um valor. Recorde-se como exemplo a função SUM, esta recebe por parâmetro um conjunto de valores que se pretendem somar, sabe-se que o resultado da aplicação dessa função ao conjunto de valores será o respectivo somatório.

Para definir o valor produzido por uma função basta no seu interior, atribuir ao nome da função um determinado valor ou expressão.

Definição do tipo de parâmetros e do tipo da função

Todos os elementos de input e output de uma função têm um tipo de dados atribuído. Assim os parâmetros deverão ser definidos com o tipo de dados respectivo e a função deverá ser definida do tipo de dados que ela envie para o exterior.

Após isto poder-se-á refinar a definição de uma função:

Function <Nome da Função> (<parametro1> **As** <Tipo>, ...) **As** <Tipo>

...

<Nome da Função> = <Valor / Expressão>

...

End Function

Nota: Se os tipos não forem definidos será assumido por defeito como sendo do tipo **Variant**

Trabalhar no Editor de VBA – Criar uma Função

Para criar uma Função é necessário que exista um módulo onde se possa escrever. Uma vez posicionado nesse módulo poderá:

- Escrever a macro integralmente, ou
- Recorrer ao Insert / Procedure para que o Visual Basic for Applications lhe crie a estrutura (Figura 15)

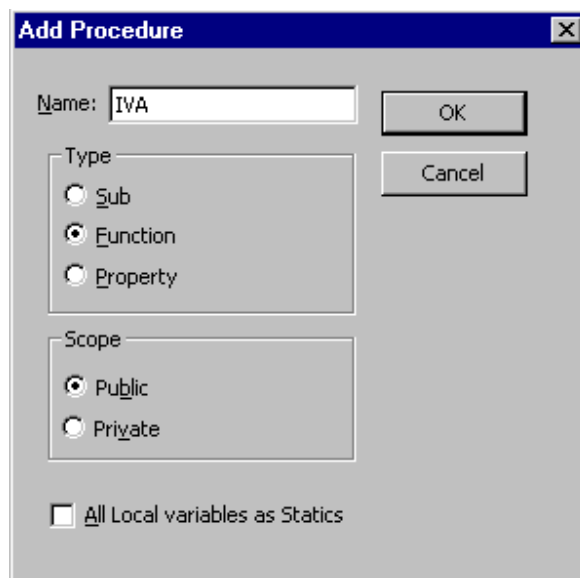


Figura 15 – Caixa de Diálogo para a criação de uma nova rotina - função

Nesta caixa de diálogo deverá indicar o tipo de rotina a criar – Function e o nome que pretende dar à função. Automaticamente o Visual Basic for Applications criará a estrutura da rotina, que neste caso ficaria:

```
Function IVA( )
```

```
End Function
```

Execução de uma Função

Uma vez que uma função produz um valor ela poderá ser executada:

- dentro de uma célula numa WorkSheet, à semelhança de qualquer uma outra função do Excel
- dentro de qualquer outra função ou SubRotina.

Com base na seguinte função, analisemos as formas de execução com mais pormenor:

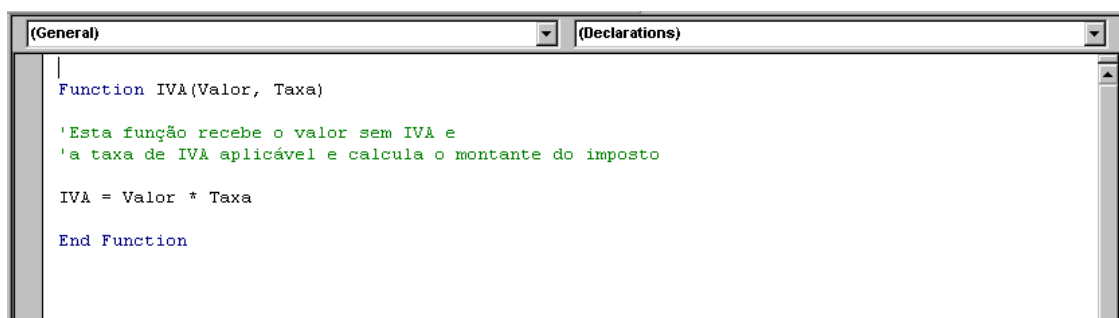


Figura 16 – Janela de Edição com a função IVA

EXECUÇÃO DENTRO DE UMA CÉLULA

1. Posicione-se na célula onde pretende inserir a função
2. Insert / Function
3. Seleccione a categoria *User Defined* – repare que aparece listada a função que acabou de criar IVA (Figura 17)

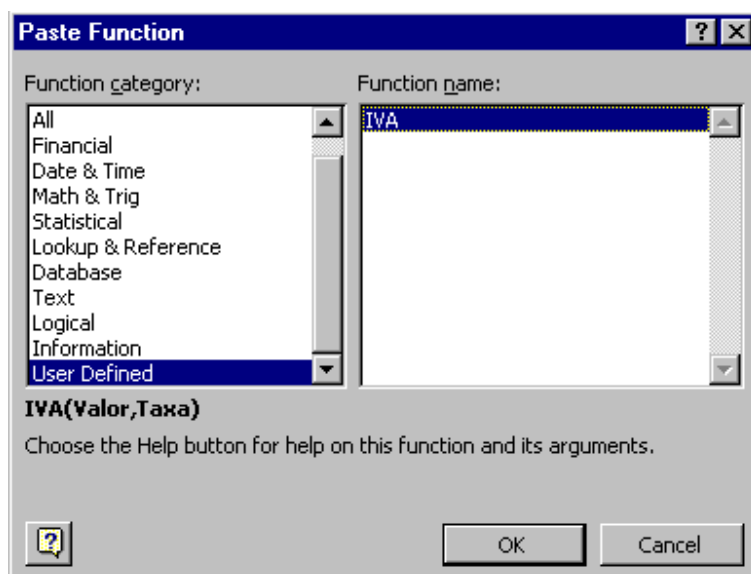


Figura 17 – Caixa de Diálogo para introdução da função

Excel – Macros e Visual Basic for Applications

4. Clique em OK e de imediato uma janela de ajuda virá auxiliar a utilização da sua função (Figura 18)

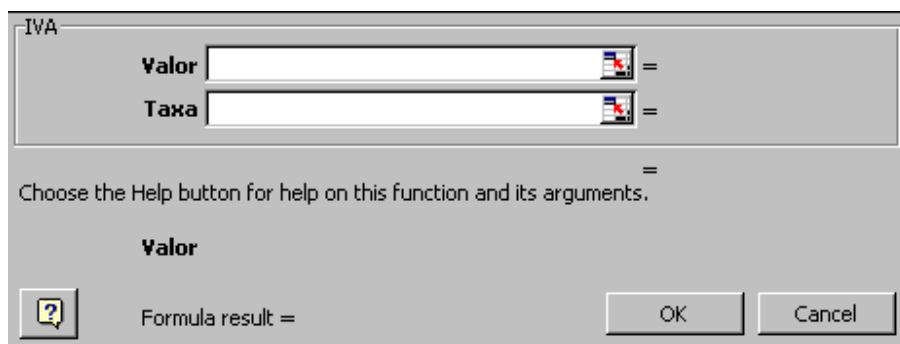


Figura 18 – Caixa de Diálogo para apoio à utilização da função

5. Introduza os parâmetros e clique em OK (Figura 19)

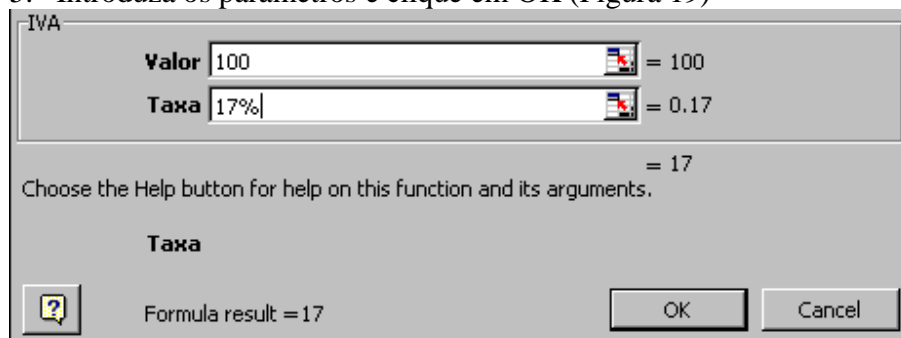


Figura 19 – Caixa de Diálogo para apoio à utilização da função – introdução de valores

Em suma:

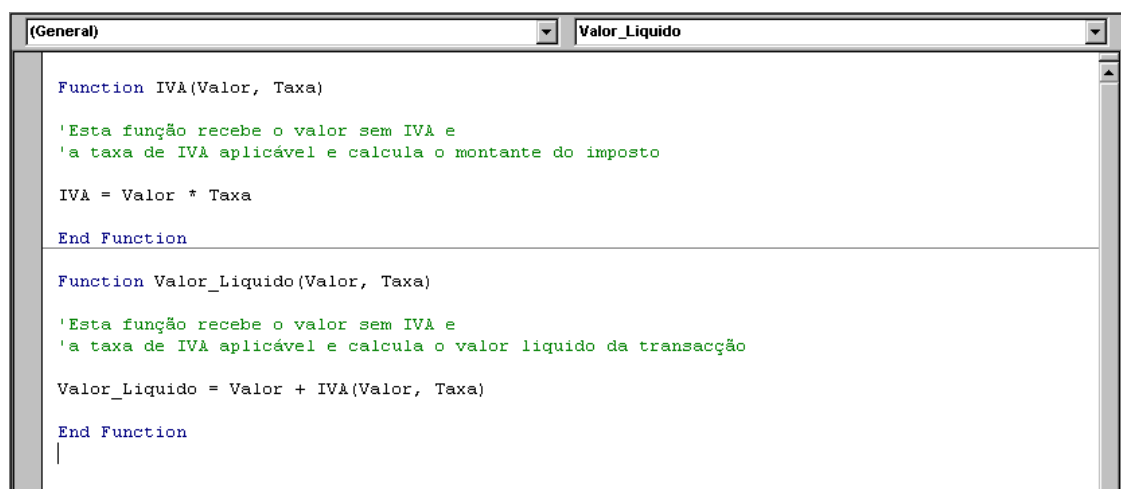
Qualquer função poderá ser chamada a partir da WorkSheet do Excel, sendo que a sua chamada será realizada à semelhança de qualquer outra função: numa célula

= <nome da função> (<Param_1> , <Param_2>)

EXECUÇÃO DENTRO DE UMA ROTINA

Quando dentro de uma rotina se faz referência ao nome de uma outra rotina a execução da primeira passa pela execução daquela que está a ser invocada.

No exemplo que se segue, a função Valor_Liquido faz uma chamada à função IVA por forma a que, o valor por esta produzido, seja adicionado à variável Valor, e assim produzir o output final da função Valor_Liquido.



```
(General) Valor_Liquido

Function IVA(Valor, Taxa)

'Esta função recebe o valor sem IVA e
'a taxa de IVA aplicável e calcula o montante do imposto

IVA = Valor * Taxa

End Function

Function Valor_Liquido(Valor, Taxa)

'Esta função recebe o valor sem IVA e
'a taxa de IVA aplicável e calcula o valor liquido da transacção

Valor_Liquido = Valor + IVA(Valor, Taxa)

End Function
```

Figura 20 – Função Valor_Liquido ao ser executada dá ordens de execução à função IVA

Um outro exemplo poderá elucidar melhor:

Excel – Macros e Visual Basic for Applications

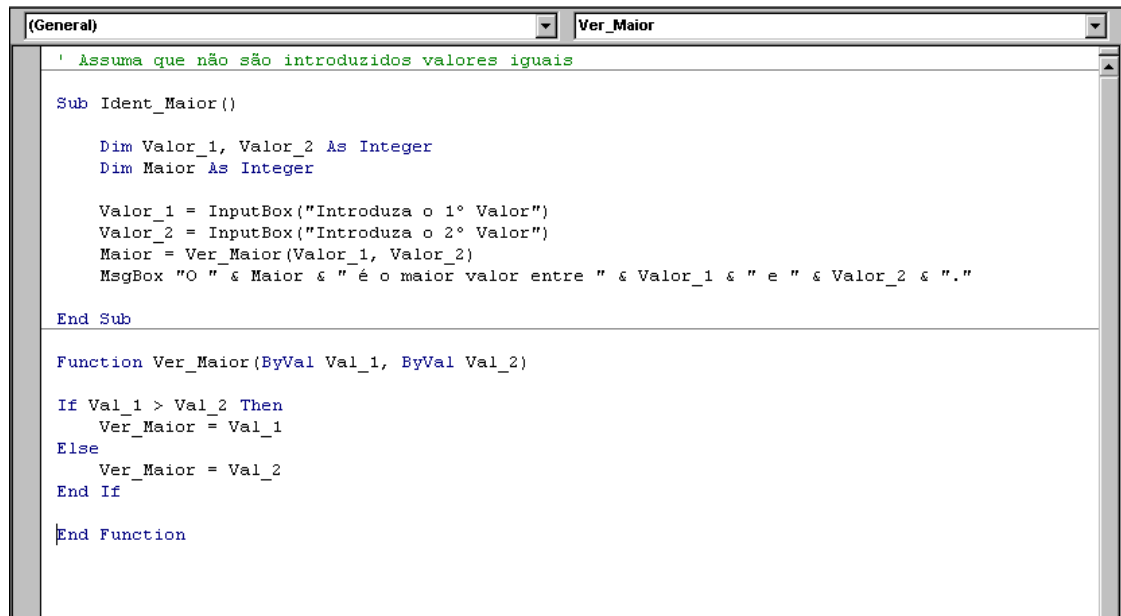


Figura 21 – A Subrotina Ident_Maior ao ser executada dá ordens de execução à função Ver_Maior

DIFERENÇAS ENTRE FUNÇÕES E ROTINAS

As funções são similares às subrotinas, existem simplesmente três diferenças:

1. As Funções Começam com a palavra-chave *Function* e terminam com as palavras *End Function*
2. As Funções podem ser chamadas a partir de fórmulas introduzidas numa WorkSheet
3. As funções retornam valores para as fórmulas ou subrotinas que as chamarem

REGRAS PARA A PASSAGEM DE PARÂMETROS

Regra 1: Como uma função retorna um valor, esta deverá ser utilizada numa expressão. Quando uma função é utilizada do lado direito de uma atribuição, ou como argumento de uma outra rotina, dever-se-á passar os parâmetros dentro de parêntesis

Regra 2: Pode-se chamar uma função ou subrotina com a palavra-chave Call, neste caso dever-se-ão colocar todos os parâmetros entre parêntesis.

Regra 3: Quando uma rotina é chamada de forma isolada e sem a palavra-chave Call, não se deve utilizar parêntesis.

Variáveis

MANUSEAMENTO COM VARIÁVEIS

O que são variáveis?

As variáveis constituem repositórios temporários de dados, podendo ser utilizadas para diversos fins.

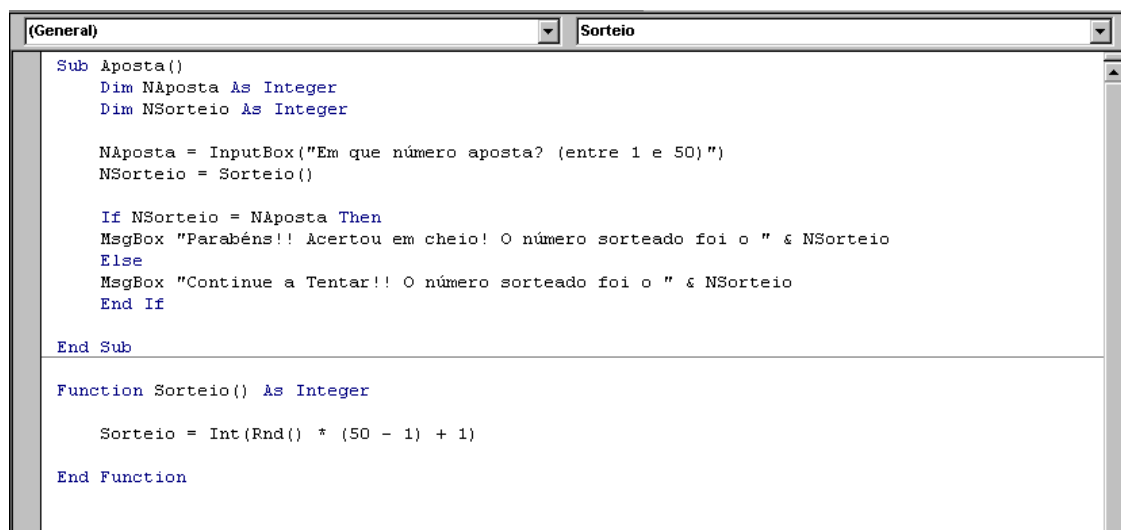


Figura 22 –Manuseamento de Variáveis

Assignação de valores a variáveis:

Excel – Macros e Visual Basic for Applications

Quando se pretende atribuir valores a variáveis dever-se-á indicar o nome da variável, o operador "=" e o valor que se pretende que a variável armazene.

<Nome_Variável> = <Valor>

Pela observação do procedimento Adição - Figura 22- podemos verificar que as variáveis *Parcela_1* e *Parcela_2* armazenam os valores introduzidos pelo utilizador através das *InputBox* (Ver capítulo "*InputBox e MsgBox*"). Também à variável *Total* vai ser atribuído o valor resultante da adição das duas parcelas - *Total = Parcela_1 + Parcela_2*.

Utilização de variáveis como se fossem valores:

O nome da variável representa o conteúdo da mesma, i.e., sempre que mencionar o nome da variável é o seu conteúdo que será considerado.

No exemplo da figura 22, pode-se constatar que na expressão:

Total = Parcela_1 + Parcela_2

Parcela_1 representa o primeiro valor introduzido e *Parcela_2* representa o segundo valor, não se trata de adicionar o nome de duas variáveis, mas adicionar os conteúdos que elas armazenam.

TIPOS DE VARIÁVEIS

Excel – Macros e Visual Basic for Applications

O tipo de variável está associado ao género de informação que esta tem hipótese de armazenar

Boolean – 2 bytes – Permite armazenar valores Booleanos – 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 objecto do Excel

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

DECLARAÇÃO DE VARIÁVEIS

Dim I_Numero As Integer

Onde:

Dim – Palavra chave que indica uma declaração de variáveis (abreviatura de dimensão)

I_Numero - 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 var1 As Integer, var2 As Date, var3 As Double

Para declarar diversas variáveis do mesmo tipo:

Dim var_1, var_2, var_3 As Currency

VARIÁVEIS – VANTAGENS DA UTILIZAÇÃO

- ◆ Simplificam a codificação, principalmente quando se necessita de utilizar um valor específico inúmeras vezes
- ◆ Com variáveis o código é mais rápido

VARIÁVEIS DO TIPO OBJECTO

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

Declaração da Variável Objecto

Dim <Var_Objecto> **As Object**

Atribuição de uma variável Objecto

Set <Var_Objecto> = <Objecto>

Onde:

Set – palavra chave que indica a atribuição de uma variável objecto

<Var_Objecto> - Variável Objecto

= - Operador de atribuição

<Objecto> - Objecto a ser atribuído à variável

Utilização Genérica da Variável Objecto

A utilização genérica do tipo Objecto serve para suportar qualquer tipo de objecto Excel (WorkBook, WorkSheet, Range,...)

Exemplo:

```
Dim Range_1 as Object  
Range_1 = Worksheet(1).Range("A1")  
Range_1.Value = 10
```

Utilização Especifica da Variável Objecto

Utiliza-se o tipo exacto do objecto que se pretende atribuir à variável. Esse tipo especifico de objectos coincide com o nome dos objectos em EXCEL.

```
Dim Range_1 As Range  
Dim WB_1 As Workbook  
Dim WS_1 As WorkSheet  
Dim XL As Aplicativo
```

Exemplo:

```
Dim Range_1 as Range  
Range_1 = Worksheet(1).Range("A1")  
Range_1.Value = 10
```

Excel – Macros e Visual Basic for Applications

Contudo estas declarações também podem ser feitas da seguinte forma genérica:

```
Dim Range_1 As Object  
Dim WB_1 As Object  
Dim WS_1 As Object  
Dim XL As Object
```

Então qual o interesse de definir as variáveis de forma específica se o podemos fazer de forma genérica?

Por uma questão de performance, se utilizar um objecto genérico, o VBA antes de executar qualquer função com o objecto tem que primeiramente o identificar (perdendo tempo) – em subrotinas simples essa diferença não é substancial mas quando se trata de grandes subrotinas já se denotam diferenças significativas.

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 de 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:

- Tools/Options
- Editor Tab
- Activar Require Variable Declaration

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

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

VARIÁVEIS – TIPOS DEFINIDOS PELO UTILIZADOR - ESTRUTURAS

Definição do Tipo

A primeira fase é constituída pela definição do tipo:

Type Dados_Pessoais

Nome **As String**

Idade **As Integer**

DataNascimento **As Date**

BI **As Long**

End Type

Criou-se um tipo de dados que representa uma estrutura com dados de diferentes tipos. Esta definição deverá ocorrer no início do módulo VBA.

Onde:

Type	Palavra-Chave que indica a definição de um tipo de dados criado pelo utilizador.
Dados_Pessoais	Nome atribuído ao tipo de dados.
Nome As String	Primeiro elemento da estrutura de dados definida.
Idade As Integer	Segundo elemento da estrutura de dados definida.
DataNascimento As Date	Terceiro elemento da estrutura de dados definida.
BI As Long	Quarto elemento da estrutura de dados definida.
End Type	Palavra-Chave que indica o fim da definição da estrutura de dados.

Utilização das Estruturas de Dados

Como utilizar as estruturas de dados:

Sub Tipos_definidos_Utilizador()

Dim Pessoa **As** Dados_Pessoais

Pessoa.Nome = “Francisco”

Pessoa.DataNascimento = #8/7/73#

Pessoa.Idade = WorksheetFunction.Year(Date)_

– WorksheetFunction.Year(Pessoa.DataNascimento)

Pessoa.BI = 103582915

MsgBox ⁵Pessoa.Nome & Chr(13) & “, Idade “ & Pessoa.Idade & Chr(13) & _

“, Data de Nascimento ” & Pessoa.DataNascimento & Chr(13) & _

“, com o BI número “_ & Pessoa.BI

End Sub

⁵ Ver capítulo *InputBox e MsgBox*

VARIÁVEIS – ARRAYS

O que é um Array ?

Um Array é uma variável que representa um conjunto de variáveis do mesmo tipo.

Os Arrays podem ser multi-dimensionais, onde todas as dimensões são indexadas numericamente.

ARRAY UNI-DIMENSIONAL

	0
	1
	2
	3
	4
	5

Um array uni-dimensional é constituído por uma única lista de elementos indexáveis. Esta lista tem um elemento inicial e um outro final sendo que a cada elemento da lista corresponde um único índice, tradução do lugar que ocupa na lista, que o identifica univocamente.

ARRAY BI-DIMENSIONAL

0	1	2	
			0
			1
			2
			3

Um array bi-dimensional é um pouco mais complexo e é constituído por um conjunto de listas do mesmo comprimento, este formato é normalmente conhecido como array ou matriz. É portanto constituída por linhas e colunas e cada elemento é identificado por um índice composto pela intercepção dos números da linha e da coluna.

Declaração de um array

ARRAYS UNI-DIMENSIONAIS

Dim ArrayNumerico(10) As Integer

Array_Numerico é o nome da variável array, o número entre parêntesis indica o número de elementos que o array pode armazenar, isto é

Array_Numerico	
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9

Em que cada elemento é do tipo Integer.

ARRAYS BI-DIMENSIONAIS

Dim Tabela_Textual (5, 4) As String

Tabela_textual é o nome da variável, os número entre parêntesis indicam que a tabela irá ter 5 linhas e 4 colunas, podendo assim armazenar 20 elementos do tipo String.

Tabela_Textual			
	0	1	2
			0
			1
			2
			3

Utilização de um Array

PARA ACEDER AO ELEMENTO

<Nome_do_Array>(<Indice1_do_Elemento>[,<Indice2_do_Elemento>,...])

ATRIBUIÇÃO DE VALORES

<Nome_do_Array>(<Indice1_do_Elemento>[,<Indice2_do_Elemento>,...]) = <Valor>

Exemplo 1:

Sub Países()

Dim Países(3) **As String**

Países (0) = "Portugal"

Países(1) = "Brasil"

Países(2) = "Moçambique"

MsgBox "Países Armazenados:" & Chr(13) & Países(0) & Chr(13) & Países(1) & _ Chr(13) & Países(2)

End Sub

Após a atribuição de valores a cada elemento do array, este passa a ter o conteúdo seguinte:

Países

Portugal	0
Brasil	1
Moçambique	2

Valor estes que serão exibidos através da MsgBox.

Excel – Macros e Visual Basic for Applications

Exemplo 2:

Option Base 1

```
Sub Utilizacao_Array()
```

```
    Dim Lotaria(3) As Integer
```

```
    Lotaria(1)=int(10000*Rnd())
```

```
    Lotaria(2)=int(10000*Rnd())
```

```
    Lotaria(3)=int(10000*Rnd())
```

```
    MsgBox "Números da lotaria: " & Lotaria(1) & ", "& Lotaria(2) & ", "& Lotaria(3)
```

```
End Sub
```

Option Base e Array Bounds

A indexação de um array por defeito tem início em 0, sendo que os respectivos índices vão de 0 a **dimensão-1**. Para alterar a base de indexação por forma a ter início em 1 basta colocar no início do módulo **Option Base 1**

Contudo se pretender que um array comece noutra número que não seja 0 ou 1, é necessário especificar os limites inferior e superior do índice aquando da declaração do array.

Exemplo:

```
Sub Utilizacao_Array()
```

```
    Dim Lotaria(4 To 5) As Integer
```

```
    Lotaria(4)=int(10000*Rnd())
```

```
    Lotaria(5)=int(10000*Rnd())
```

```
    MsgBox "Números da lotaria: " & Lotaria(4) & ", "& Lotaria(5)
```

```
End Sub
```

Excel – Macros e Visual Basic for Applications