Visual Basic for Applications

- ***Introdução***
  - É uma linguagem de programação baseada na conhecida linguagem BASIC
  - Está concebida para funcionar em conjunto com diferentes aplicações, de forma a potenciar a robustez das mesmas
  - Enquadra-se nos ambientes de programação baseados no processamento de sequência de eventos (*event-driven programming*)
- ***História***
  - Foi inicialmente integrada com o Excel 5 em 1994 e a partir daí a sua expansão para outras aplicações foi gradual
  - Foi com a saída do Office 97 em 1997 que a Microsoft concretizou um dos seus grandes objectivos: ter um ambiente de programação completamente integrado nos seus quatro produtos mais famosos: Word, Excel, Access e PowerPoint
  - Actualmente, o VBA é já por si só um produto independente, que outras companhias podem adoptar e incorporar nas suas aplicações

Variáveis I

- ***Para que servem?***
  - Servem para guardar valores temporariamente em memória
- ***Exemplo: dado X quanto é a sua metade mais o seu dobro?***

```
metade = x / 2
dobro = 2 * x
resultado = metade + dobro
```

- ***Declaração explícita de variáveis***
  - Declarar uma variável **VAR**: `Dim VAR`
  - Declarar uma variável **VAR** como sendo do tipo **TYPE**: `Dim VAR As TYPE`
  - Variáveis sem declaração de tipo têm por defeito o tipo `Variant`
- ***Declaração implícita de variáveis***
  - Possibilidade de não declarar variáveis
  - Variáveis não declaradas têm por defeito o tipo `Variant`
  - Não permitir o uso de variáveis implícitas: `Option Explicit`

Variáveis II

- ***Tipo de variáveis***
  - **`Variant`** tipo genérico
  - **`Boolean`** True ou False
  - **`Byte`** 0 até 255
  - **`Integer`** -32.768 até 32.767
  - **`Long`** -2.147.483.648 até 2.147.483.647
  - **`Decimal`** +/-79.228.162.514.264.337.593.543.950.335 (sem casas decimais)
    +/-7,9228162514264337593543950335 (com casas decimais)
  - **`Currency`** -922.337.203.685.477,5808 até 922.337.203.685.477,5807
  - **`Date`** 1 de Janeiro de 100 até 31 de Dezembro de 9999
  - **`String`** 1 até aproximadamente 2 biliões de caracteres

Variáveis III

- *Exemplos*

```
Option Explicit

Dim metade As Decimal
Dim dobro As Decimal
Dim resultado As Decimal
metade = x / 2
dobro = 2 * x
resultado = metade + dobro

Dim val As Boolean
val = True

Dim texto As String
texto = “vba”

Dim aux As Variant
aux = True
aux = “vba”
```

Operadores I

- ***Aritméticos***
  - `+` (adição)
  - `-` (subtracção e negação)
  - `*` (multiplicação)
  - `/` (divisão)
  - `\` (divisão inteira)
  - `^` (exponenciação)
  - `Mod` (resto da divisão)
- ***Texto***
  - `&` (concatenação)
- ***Comparação***
  - `=` (igual a)
  - `>` (maior que)
  - `<` (menor que)
  - `<>` (diferente de)
  - `>=` (maior ou igual)
  - `<=` (menor ou igual)
- ***Lógicos***
  - `And` (e lógico)
  - `Or` (ou lógico)
  - `Not` (negação)

Operadores II

- ***Precedências***

Maior precedência

`^`	exponenciação
`-`	negação
`*` , `/`	multiplicação e divisão
`\`	divisão inteira
`Mod`	resto da divisão
`+` , `-`	adição e subtracção
`&`	concatenação
`=` , `>` , `<` , `<>` , `>=` , `<=`	comparação
`And` , `Or`, `Not`	lógicos

Menor precedência

- ***Outros caracteres***

`'`	comentários
`:`	múltiplas instruções na mesma linha
`_`	uma instrução em múltiplas linhas

Caixas de mensagem

- ***Caixas MsgBox***
  - Forma básica de apresentar uma mensagem ao utilizador
  - `MsgBox(mensagem)`

- ***Caixas InputBox***
  - Apresenta uma mensagem numa caixa de mensagem, permite que o utilizador introduza texto, e devolve a sequência de texto introduzida
  - `InputBox(mensagem)`

- ***Exemplo***

```
Dim nome As String
nome = InputBox(“Introduza o seu nome...”)
MsgBox “Bem vindo ” & nome & “!”
```

Procedimentos I

- ***Para que servem?***
  - Quando uma dada sequência de instruções (tarefa) é executada repetidamente em diferentes partes do código, deve ser criado um procedimento que substitua e concentre num único local a sequência de instruções da tarefa a executar
  - O uso de procedimentos aumenta a produtividade do programador pois diminui o tamanho global do código a escrever, facilita a edição da sequência de instruções relativa à tarefa em causa, e minimiza a ocorrência de erros

- ***Tipo de procedimentos***
  - **`Sub`**: são utilizados para executar tarefas independentes
  - **`Function`**: são utilizados para calcular/retornar valores

Procedimentos II

- ***Procedimentos Sub***

```
Sub nome ([argumentos])
  [...]
  [Exit Sub]
  [...]
End Sub
```

- ***Exemplo***

```
Sub boas_vindas()
  Dim nome As String
  nome = InputBox("Introduza o seu nome...")
  MsgBox "Bem vindo " & nome & "!"
End Sub
```

Procedimentos III

- **Procedimentos Function**

```
Function nome ([argumentos]) [As tipo]
   [...]
   [nome = expressão]
   [...]
   [Exit Function]
   [...]
   [nome = expressão]
   [...]
End Function
```

- **Exemplo**

```
Function area(comp As Integer, alt As Integer) As Integer
  area = comp * alt
End Function
```

Argumentos I

- ***Declarar argumentos***
  - **argumento [As tipo]**
- ***Exemplo***

```
Function area(comp As Integer, alt As Integer) As Integer
```

- ***Passar e nomear argumentos***
  - Passar argumentos: `area(5, 4)`
  - Nomear argumentos: `area (alt:= 4, comp:= 5)`
- ***Argumentos opcionais***
  - **Optional argumento [As tipo] = valor_por_defeito**
  - Permite atribuir um valor por defeito nos casos em que não é passado ou nomeado qualquer valor
  - A declaração de um argumento opcional implica que os argumentos subsequentes sejam igualmente declarados como opcionais

Argumentos II

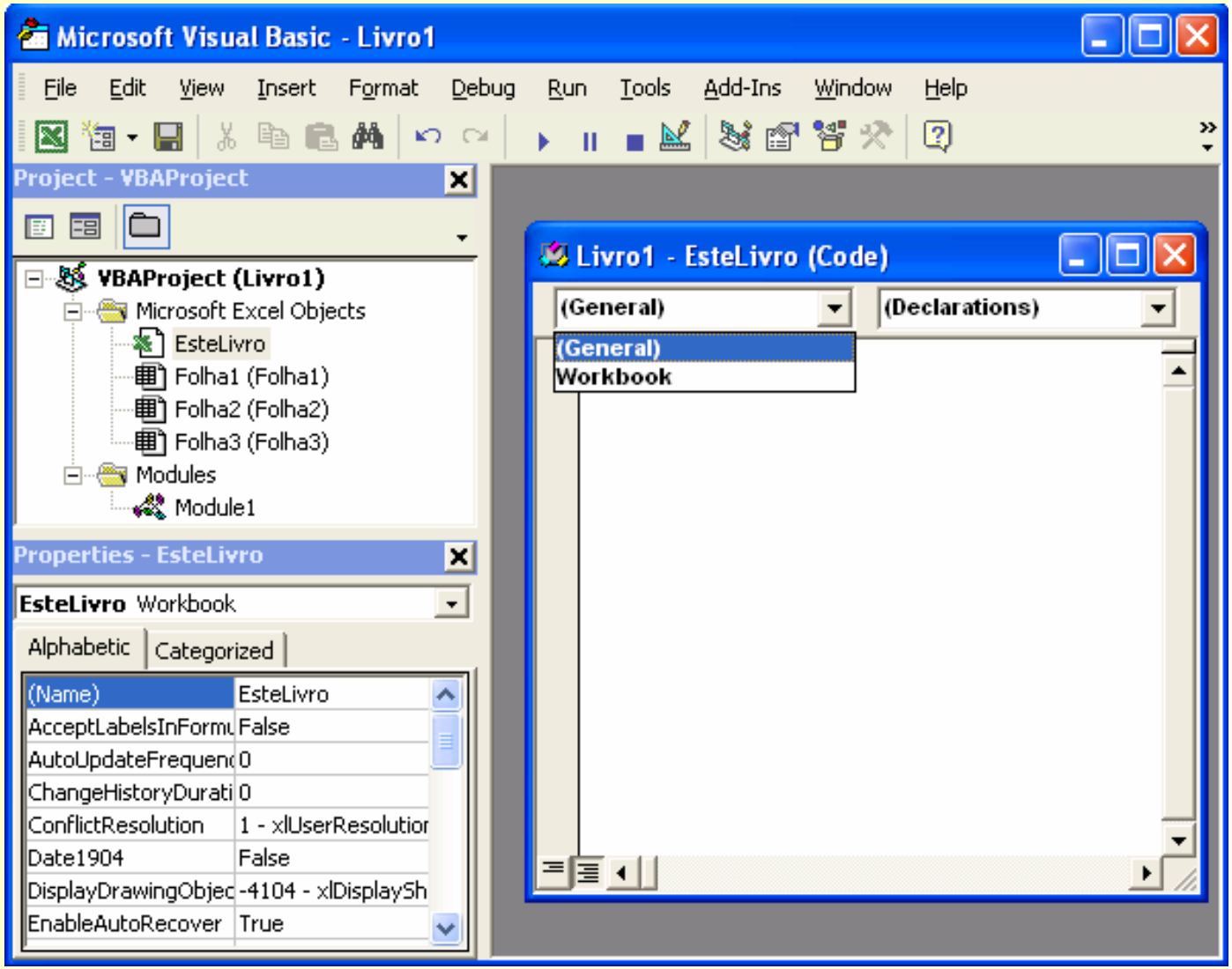
- *Exemplos*

```
Function area(Optional comp As Integer = 1,
              larg As Integer = 1)
  area = comp * larg
End Function                                    'dá um erro



Function area(Optional comp As Integer = 1,
              Optional larg As Integer = 1)
  area = comp * larg
End Function


aux = area (5, 4)                               'aux = 20
aux = area (5)                                  'aux = 5
aux = area ()                                   'aux = 1
```

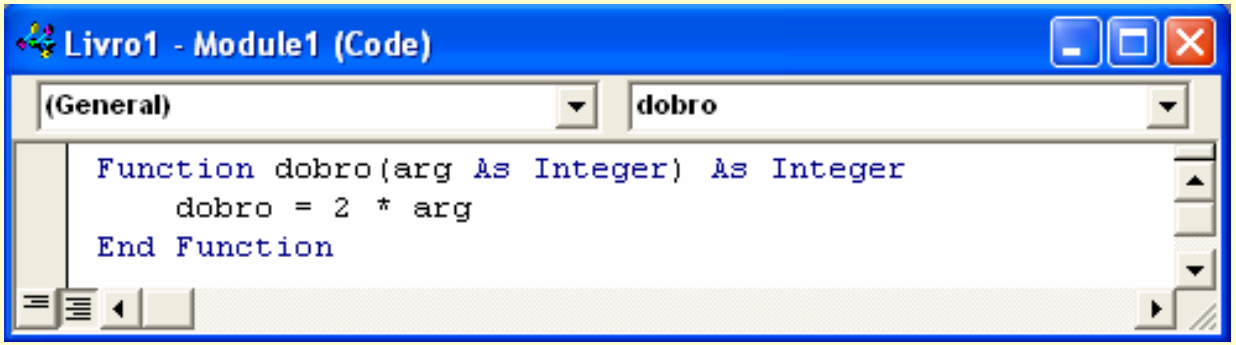
VBA e Excel

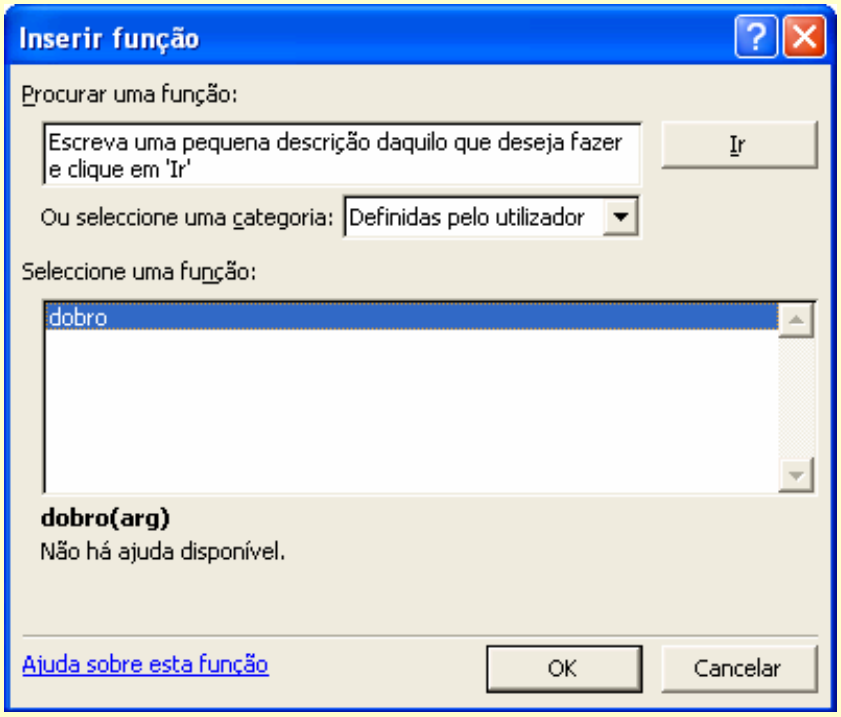
- ***Editor do Visual Basic***
  - Menu <Ver> seguido das opções <Barras de ferramentas> e <Visual Basic>

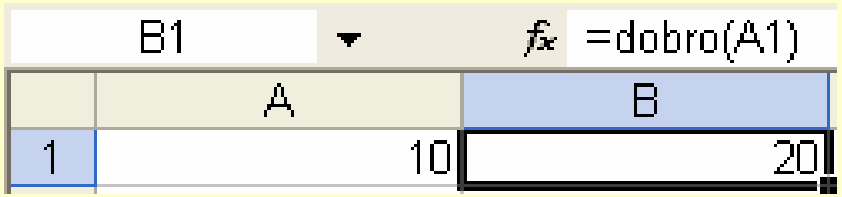
  - Janelas de projecto, de propriedades e de código



Código VBA como função do Excel







Execução condicional I

- ***If Then Else***

```
If condição_1 Then
  [...]
...
[ElseIf condição_n Then
  [...]]
[Else
  [...]]
End If
```

- ***Exemplos***

```
Function escala(num As Integer) As String
  If num = 0 Then
    escala = “zero”
  ElseIf num < 0 Then
    escala = “negativo”
  Else                                  ‘num > 0
    escala = “positivo”
  End If
End Function
```

Execução condicional II

- ***Exemplos***

```
Function é_positivo(num As Integer) As Boolean
  If num > 0 Then
    é_positivo = True
  Else                                  'num <= 0
    é_positivo = False
  End If
End Function


Function num_min(num1 As Integer, num2 As Integer)
  If num1 < num2 Then
    num_min = num1
  Else                                  'num1 >= num2
    num_min = num2
  End If
End Function
```

Execução condicional III

- **Exemplos**

```
Function num_meio(num1 As Integer, _
                  num2 As Integer, _
                  num3 As Integer) As Integer
  If num1 >= num2 And num2 >= num3 Then
    num_meio = num2
  ElseIf num1 >= num3 And num3 >= num2 Then
    num_meio = num3
  ElseIf num2 >= num1 And num1 >= num3 Then
    num_meio = num1
  ElseIf num2 >= num3 And num3 >= num1 Then
    num_meio = num3
  ElseIf num3 >= num1 And num1 >= num2 Then
    num_meio = num1
  ElseIf num3 >= num2 And num2 >= num1 Then
    num_meio = num2
  End If
End Function
```

Ciclos numeráveis I

- ***For Next***

```
For contador = início To fim [Step incremento]
   [...]
   [Exit For]
   [...]
Next
```

- ***Condições de paragem***
  - **`Step`**: por defeito o valor de **`incremento`** é 1
  - Se o **`incremento`** for positivo ou zero, o ciclo termina assim que **`contador`** seja maior do que **`fim`**
  - Se o **`incremento`** for negativo, termina assim que **`contador`** seja menor do que **`fim`**

Ciclos numeráveis II

- **Exemplos**

```
Function mult_int(x As Integer, y As Integer) As Integer
  Dim contador As Integer
  If x >= 0 Then
    mult_int = 0
    For contador = 1 To x
      mult_int = mult_int + y
    Next
  Else
    mult_int = 0
    For contador = -1 To x Step -1
      mult_int = mult_int - y
    Next
  EndIf
End Function
```

Ciclos numeráveis III

- *Exemplos*

```
Function mult_int(x As Integer, y As Integer) As Integer
  Dim contador As Integer
  If x >= 0 Then
    mult_int = 0
    For contador = 1 To x
      mult_int = mult_int + y
    Next
  Else
    mult_int = -mult_int(-x,y)
  EndIf
End Function

Function soma_pares(limite As Integer) As Long
   Dim contador As Integer
  soma_pares = 0
  For contador = 2 To limite Step 2
    soma_pares = soma_pares + contador
   Next
End Function
```

Ciclos numeráveis IV

- ***Exemplos***

```
Function primo(num As Long) As Boolean
  Dim contador As Long
  For contador = 2 To num \ 2
    If num Mod contador = 0 Then
      primo = False
      Exit Function
    End If
  Next
  primo = True
End Function
```

Ciclos condicionais I

- ***Do Loop***

```
Do {While | Until} condição
   [...]
   [Exit Do]
   [...]
Loop

Do
   [...]
   [Exit Do]
   [...]
Loop {While | Until} condição
```

- ***Condições de paragem***
  - **`While`**: executa o ciclo enquanto a condição for verdade
  - **`Until`**: executa o ciclo enquanto a condição for falsa

Ciclos condicionais II

- *Exemplos*

```
Function mult_int(x As Integer, y As Integer) As Integer
  If x >= 0 Then
    mult_int = 0
    Do While x > 0
      mult_int = mult_int + y
      x = x – 1
    Loop
  Else
    mult_int = -mult_int(-x,y)
  EndIf
End Function

Sub confirmar_password()
   Dim password As String
  Do
    password = InputBox(“Insira a password de acesso...”)
  Loop While password <> “cta”          ‘Until password = “cta”
End Sub
```

Manipulação de strings I

- ***Funções básicas***
  - `Len(string)`
  - `LCase(string)`
  - `UCase(string)`
  - `Left(string, comprimento)`
  - `Right(string, comprimento)`
  - `Mid(string, início [, comprimento])`
- ***Exemplos***

```
Len("Hello World")                    '11
Lcase("Hello World")                  '"hello world"
Ucase("Hello World")                  '"HELLO WORLD"
Left("Hello World", 1)                '"H"
Right("Hello World", 3)               '"rld"
Mid("Hello World", 7)                 '"World"
Mid("Hello World", 7, 2)              '"Wo"
```

Manipulação de strings II

- ***Funções básicas***
  - `LTrim (string)`
  - `RTrim (string)`
  - `Trim (string)`
  - `InStr ([início,] string_geral, string_procura)`
  - `StrComp (string1, string2)`
- ***Exemplos***

```
LTrim(“  <-  ->  ”)                    ‘“<-  ->  ”
RTrim(“  <-  ->  ”)                    ‘”  <-  ->”
Trim(“  <-  ->  ”)                     ‘“<-  ->”
InStr(1, “Hello World”, “o”)           ‘5
InStr(6, “Hello World”, “o”)           ‘8
StrComp(“abc”, “abc”)                  ‘0 (string1 = string2)
StrComp(“abc”, “ABC”)                  ‘1 (string1 > string2)
StrComp(“ABC”, “abc”)                  ‘-1 (string1 < string2)
```

Manipulação de strings III

- ***Exemplo***

```
Function inverte(str As String) As String
  Dim i As Integer
  Dim tamanho As Integer
  tamanho = Len(str)
  inverte = ""
  For i = tamanho To 1 Step -1
    inverte = inverte & Mid(str, i, 1)
  Next
End Function
```

Manipulação de datas e horas I

- ***Funções básicas***
  - `Date`
  - `Time`
  - `Now`
  - `Year(data)`
  - `Month(data)`
  - `Day(data)`
  - `Hour(hora)`
  - `Minute(hora)`
  - `Second(hora)`
  - `DateSerial(ano, mês, dia)`
  - `TimeSerial(hora, minuto, segundo)`

Manipulação de datas e horas II

- ***Exemplos***

```
Date                                ‘27-11-2003
Time                                ‘11:20:00
Now                                 ‘27-11-2003 11:20:00
Year(Date)                          ‘2003
Hour(Time)                          ‘11
DateSerial(2003, 11, 27)            ‘27-11-2003
TimeSerial(11, 20, 0)               ‘11:20:00
```

```
Function dias_de_vida (dia As Integer, mes As Integer, _
                       ano As Integer)
  Dim nascimento As Date
  nascimento = DateSerial(ano, mes, dia)
  dias_de_vida = Date - nascimento
End Function
```

Objectos I

- ***Conceito***
  - Qualquer coisa que numa aplicação se pode manipular de algum modo
- ***Formas de manipular um objecto***
  - Alterar o seu conjunto de ***propriedades***
  - Activar ***métodos*** específicos do objecto para executar determinadas tarefas
  - Associar procedimentos aos ***eventos*** que podem ocorrer sobre o objecto
- ***Propriedades***
  - As propriedades são os atributos que definem as características dos objectos
    - `Cells(1,1).Formula = “=B1*10”`
  - Certas propriedades são elas mesmas objectos
    - `Cells(1,1).Font.Italic = True`
  - Existem propriedades que são só de leitura
    - `col = Cells(1,1).Column`

Objectos II

- ***Métodos***
  - Os métodos são acções que descrevem o que os objectos podem fazer
  - São executados sobre os objectos e podem conter ou não argumentos
    - `Cells(1,1).Clear`
    - `Cells(1,1).AddComment “comentário”`

- ***Eventos***
  - Os eventos são algo que acontece aos objectos
  - Ocorrem em resultado de acções do utilizador, do sistema ou do próprio código
  - É possível associar a execução de procedimentos à ocorrência de eventos
    - `Private Sub BotãoOK_Click()`
  - Os procedimentos correspondentes aos eventos têm todos a seguinte forma:
    - `Private Sub Objecto_Evento(argumentos)`

Objectos Range I

- ***Referenciação***
  - **`Range("A1")`**: referência à célula A1
  - **`Range("B2:C3")`**: referência ao intervalo B2:C3
  - **`Range("A1, B2:C3")`**: referência à célula A1 mais o intervalo B2:C3
  - **`Cells(1, 2)`**: referência à célula na 1ª linha e na 2ª coluna (célula B1)
  - **`Range("B2:C3").Cells(1, 2)`**: referência à célula na 1ª linha e na 2ª coluna dentro do intervalo B2:C3 (célula C2)
- ***Exemplo***

```
For ano = 1 To 100
  Range("A" & ano) = 1999 + ano
Next

For mes = 1 To 12
  Cells(1, mes) = mes
Next
```

Objectos Range II

- ***Fórmulas***
  - **range.Formula**: fórmulas das células do *range no idioma standard (inglês)*
  - **range.FormulaLocal**: fórmulas das células do *range* no idioma local
- ***Exemplos***

```
Range("A1").Formula = "=A2"
Range("A9").Formula = "=SUM(A1:A8)"
Range("A9").FormulaLocal = "=SOMA(A1:A8)"

'aspas nas fórmulas devem ser duplicadas
Range("A1").Formula = "=COUNTIF(B1:B9, ""0"")"
Range("A1").FormulaLocal = "=CONTAR.SE(B1:B9; ""0"")"

'as fórmulas actualizam-se tal como quando são arrastadas
'o exemplo que se segue leva a que B9 ← =SOMA(B1:B8)
Range("A9:B9").FormulaLocal = "=SOMA(A1:A8)"
```

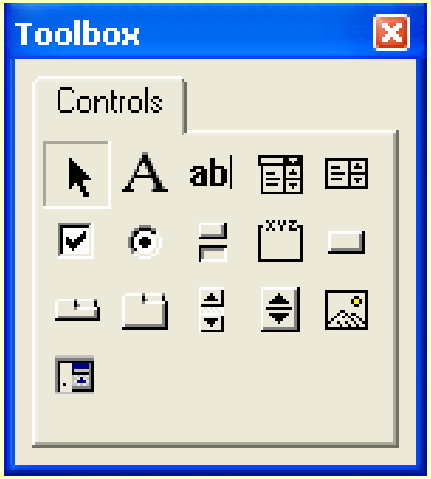
Objectos Range III

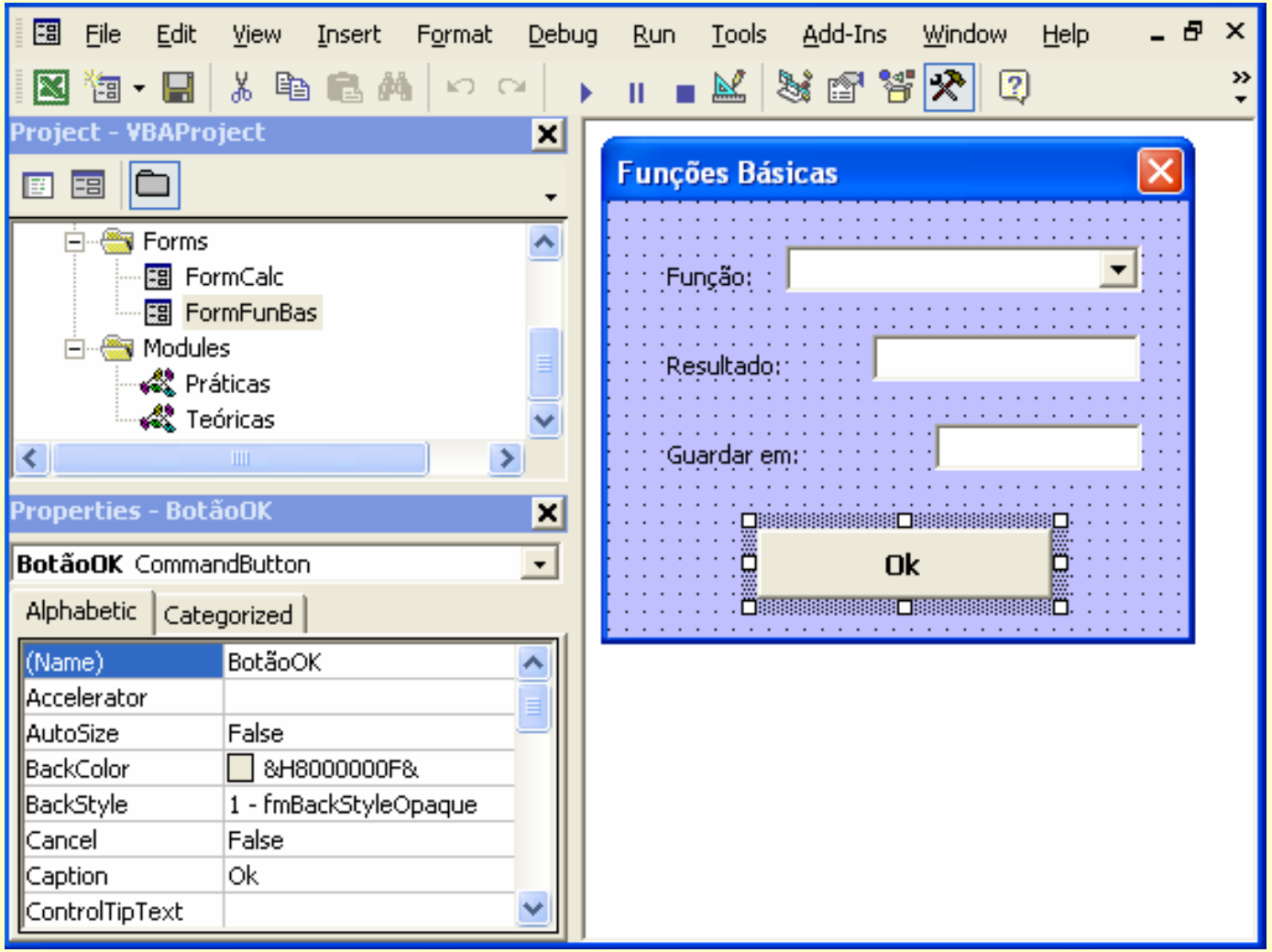
- *Exemplo*

```
Sub colunaA_azul(inicio As Integer, fim As Integer)
  Dim linha As Integer
  For linha = inicio To fim
    Cells(linha, 1).Interior.ColorIndex = 5   ‘5 é a cor azul
  Next
End Sub


Sub colunaA_azul(inicio As Integer, fim As Integer)
    Range(“A” & inicio & “:A” & fim).Interior.ColorIndex = 5
End Sub
```

Formulários e Controlos

- **Formulários**
  - Menu <Inserir> seguido da opção <UserForm>
- **Controlos**
  - Caixa de ferramentas





Objectos Control I

- ***Propriedades comuns***
  - **`Name`**: nome que identifica o controlo
  - **`Left`** / **`Top`**: posição relativa ao canto superior esquerdo do formulário
  - **`Height`** / **`Width`**: altura e comprimento do controlo
  - **`BackColor`** / **`ForeColor`**: cor do fundo e cor do texto do controlo
  - **`Font`**: tipo de letra do controlo
  - **`Caption`**: texto presente no controlo
  - **`ControlTipText`**: texto de ajuda para quando se mantém o foco sobre o controlo
  - **`TabIndex`**: ordem de navegação do controlo (utilizando a tecla *Tab)*
  - **`TabStop`**: se **`False`** previne a navegação por intermédio da tecla *Tab*
  - **`Visible`**: visibilidade do controlo
  - **`Enabled`**: se **`False`** previne o controlo de receber o foco e responder a eventos
  - **`Locked`**: se **`True`** previne o utilizador de editar o valor presente no controlo

Objectos Control II

- ***Command button***
  - **`CommandButton_Click()`**: ocorre quando o botão é premido
- ***Text box***
  - **`TextBox_Change()`**: ocorre quando o conteúdo da caixa é alterado
- ***Combo / List boxes***
  - **`.AddItem`**: adiciona uma nova entrada à caixa
  - **`.ListIndex`**: entrada seleccionada na caixa (a primeira entrada é a zero)
  - **`ComboBox_Change()`**: ocorre quando o conteúdo da caixa é alterado
- ***UserForm***
  - **`UserForm_Initialize()`**: ocorre quando o formulário é iniciado

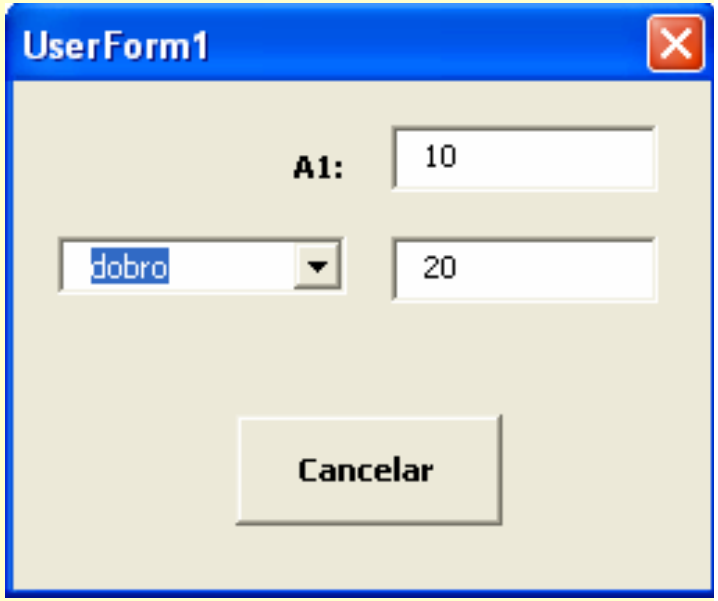
Objectos Control III

- ***Exemplo***
  - O formulário ao lado possui duas caixas de texto de nomes **`CaixaValor`** e **`CaixaResultado`**, uma caixa de combinação de nome **`ListaFunções`** e um botão de comando de nome **`BotãoCancelar`**. Defina os procedimentos necessários para obter a seguinte funcionalidade:

  - O activar do formulário deve copiar para **`CaixaValor`** o conteúdo da célula A1 e deve iniciar **`ListaFunções`** com os itens dobro e triplo
  - A selecção de um item em **`ListaFunções`** deve colocar em **`CaixaResultado`** o resultado da aplicação do item seleccionado sobre o valor em **`CaixaValor`**. Por exemplo, se **`CaixaValor`** apresentar o valor 10 e o item seleccionado for dobro, então **`CaixaResultado`** deverá ficar com o valor 20
  - O pressionar do **`BotãoCancelar`** deve fechar o formulário

Objectos Control IV

- *Exemplo*




```
Private Sub UserForm_Initialize()
  CaixaValor = Range("A1")
  ListaFunções.AddItem "dobro"
  ListaFunções.AddItem "triplo"
End Sub

Private Sub ListaFunções_Change()
  If CaixaValor = "" Then
    CaixaResultado = ""
  ElseIf ListaFunções.ListIndex = 0 Then
    CaixaResultado = 2 * CaixaValor
  ElseIf ListaFunções.ListIndex = 1 Then
    CaixaResultado = 3 * CaixaValor
  End If
End Sub

Private Sub BotãoCancelar_Click()
  Unload UserForm1                    'fecha o formulário
End Sub
```