

Mestrado em Métodos Quantitativos para a Decisão
Económica e Empresarial

Computação

Filipe Rodrigues

Gab. 302 Quelhas 2

frodrigues@iseg.ulisboa.pt

Avaliação

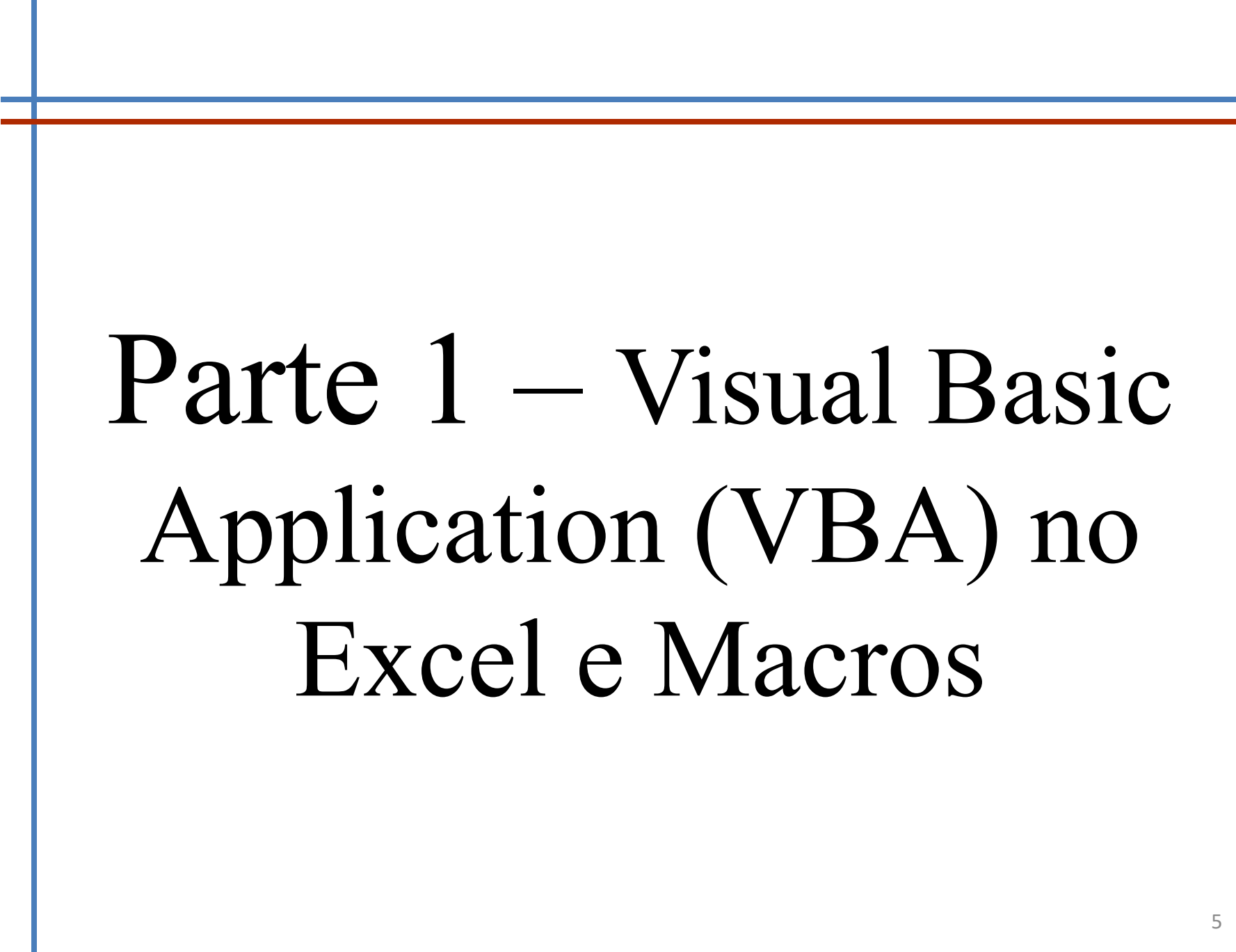
- **Primeiro teste VBA (20%)**
22 de setembro
- **Segundo teste VBA (30%)**
20 de outubro
- **Projeto (5+8+17%)**
Ao longo do semestre...
- **Teste Python (20%)**
4 de dezembro

Bibliografia

- ✓ *Introduction to Algorithms*, T. Cormen, C. Leiserson, R. Rivest and C. Stein (2001), 2nd ed., MIT, Massachusetts.
- ✓ *Algoritmia e Estruturas de Dados*, J.B. Vasconcelos e J.V. Carvalho (2005), Centro Atlântico.
- ✓ *Mastering VBA for Microsoft Office 365*, R. Mansfield, Sybex; 2019 edition 2019.
- ✓ *Developing Spreadsheet-Based Decision Support Systems*, S. D. Eksioglu, M. Seref, R. Ahuja and W. Winston, Dynamic Ideas LLC; 2nd edition, 2011.

Capítulo I

Introdução à Programação (VBA)

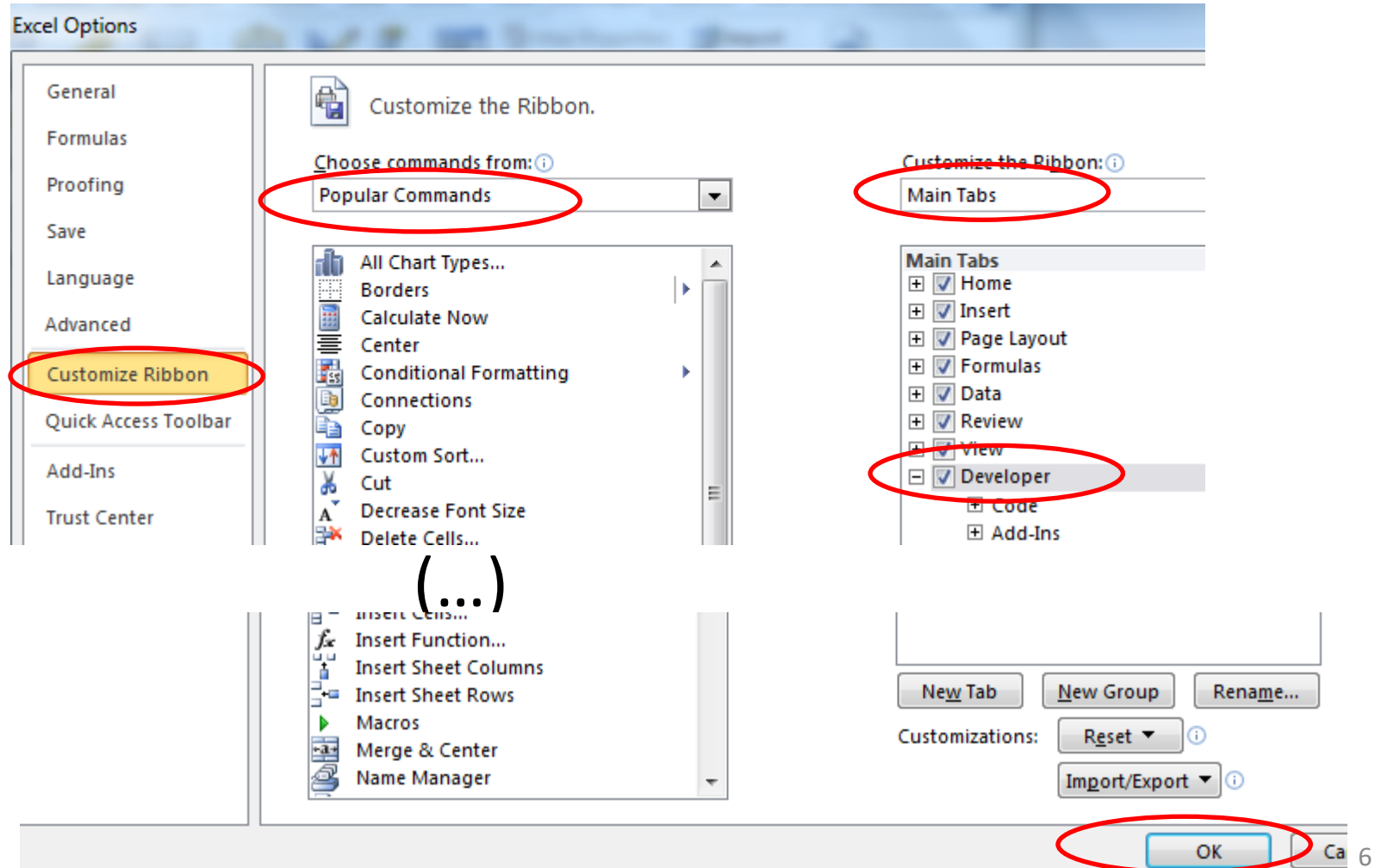


Parte 1 – Visual Basic Application (VBA) no Excel e Macros

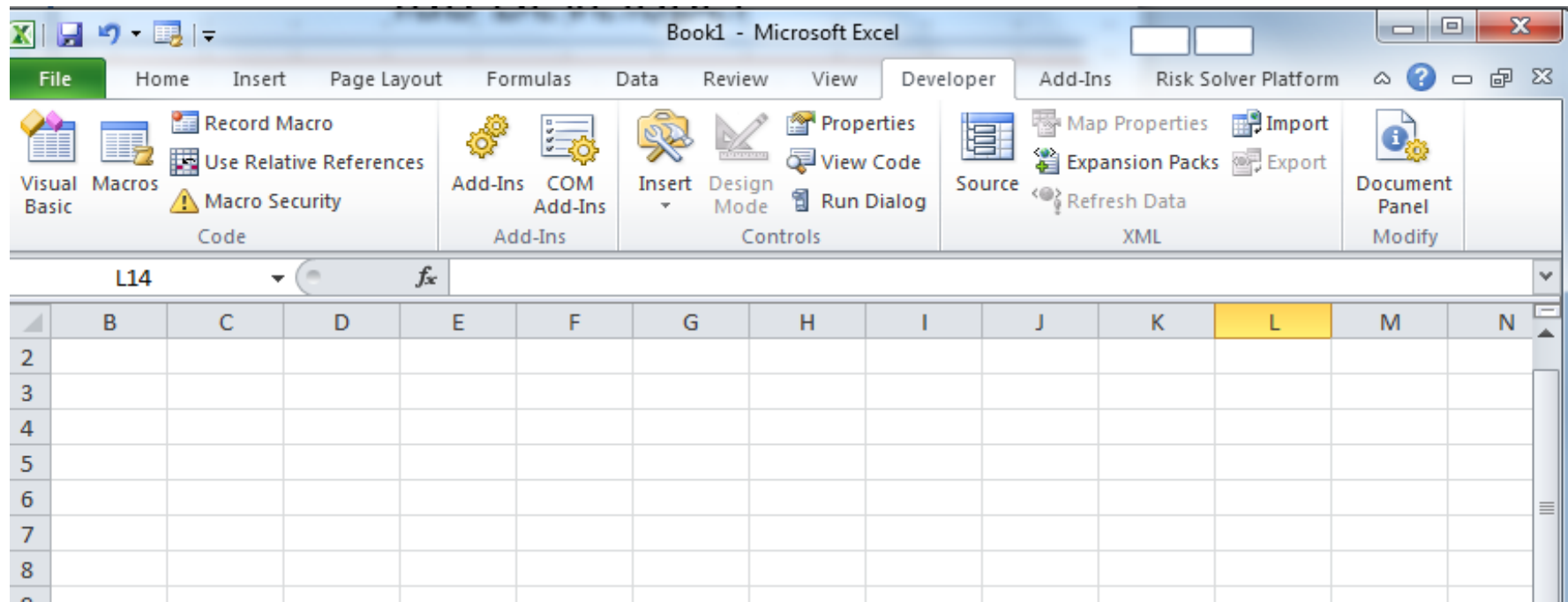
VBA acessivel no Excel (2010)

1. File -> Options

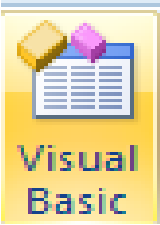
2.



Tab Developer

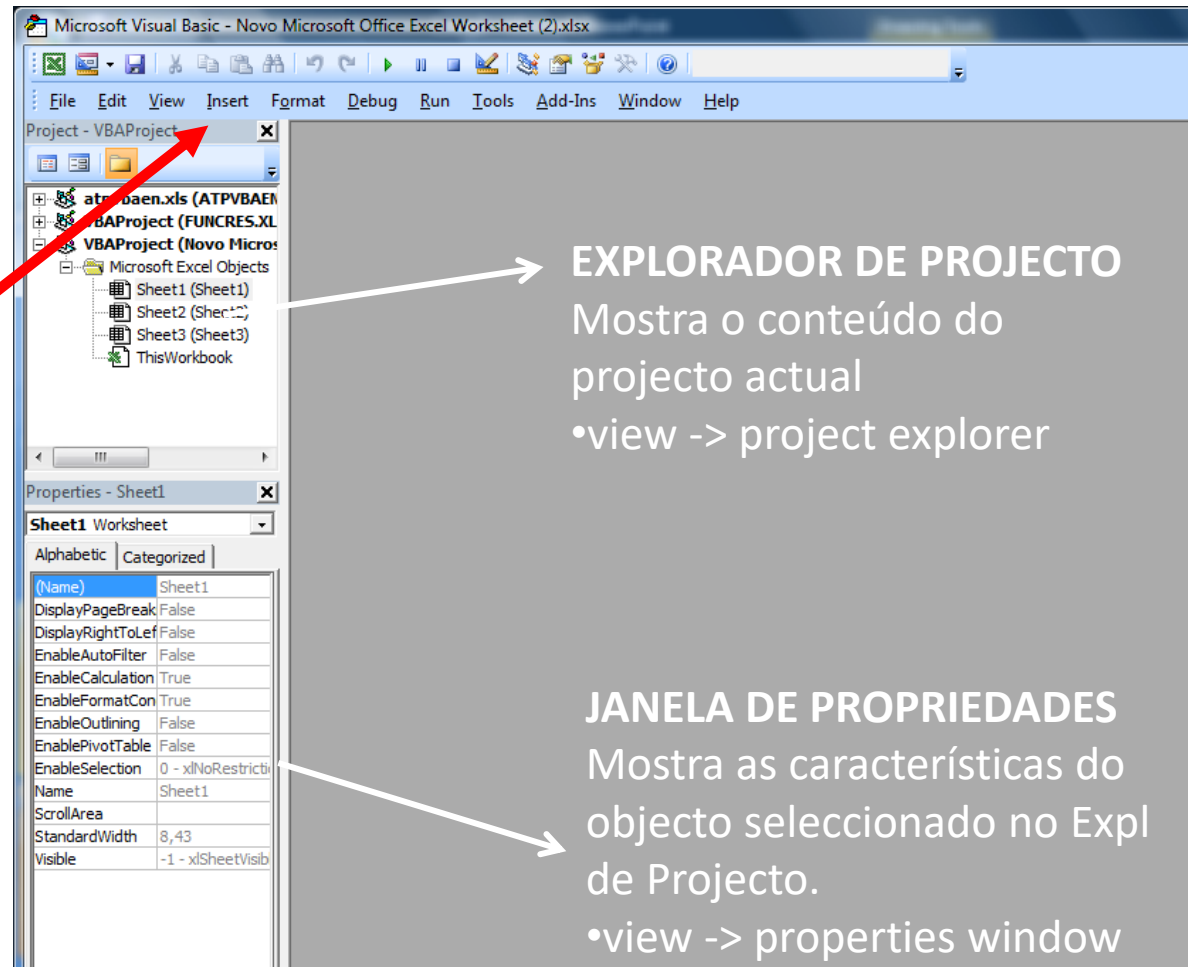


O editor do VBA

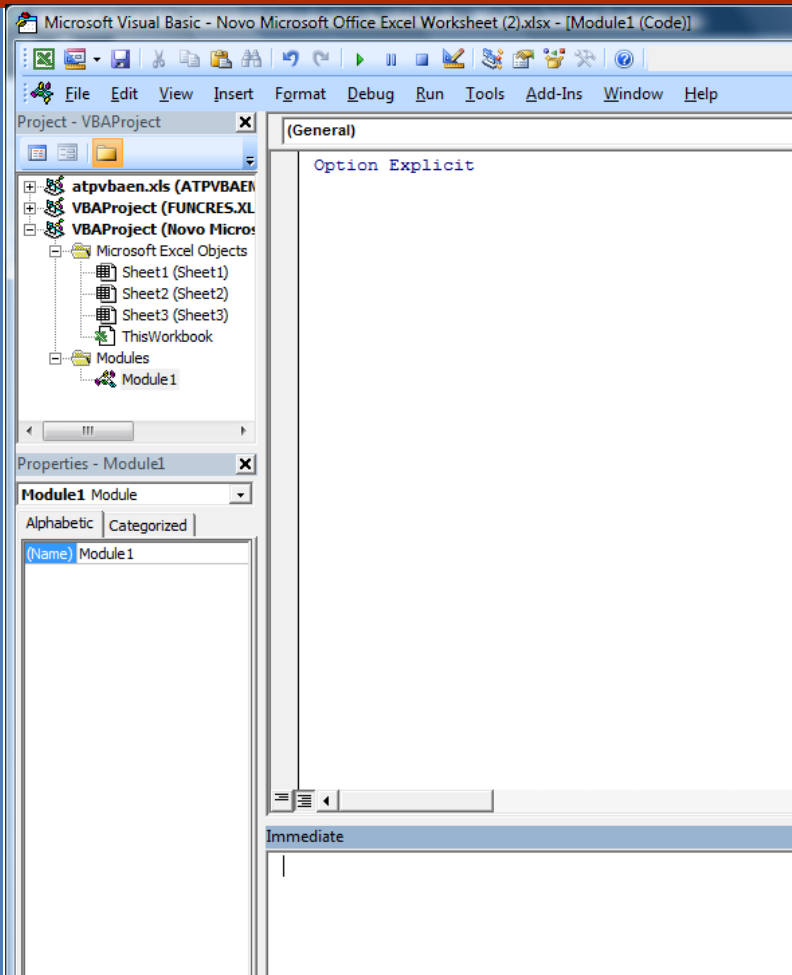
- IDE (Integrated Development Environment)
 - Para aceder ao IDE a partir do Excel
 - (ALT+F11) ou
 - Developer ->
- 
- The icon is a yellow square with a blue border. Inside, there is a blue document icon with a yellow tab. Below the document icon, the words "Visual Basic" are written in blue, stacked vertically.
- Para voltar à folha de cálculo
 - (ALT+F11) (novamente)

O editor do VBA

Abrir novo
editor de
código:
Insert/module



O editor do VBA



Macro

- É um programa escrito em VBA, que indica a uma aplicação como o Excel quais os passos a executar para atingir um objetivo específico. Dito de outro modo, é a descrição formalizada das tarefas que se pretende automatizar.
- Serve para automatizar tarefas repetitivas.

Objetos gráficos

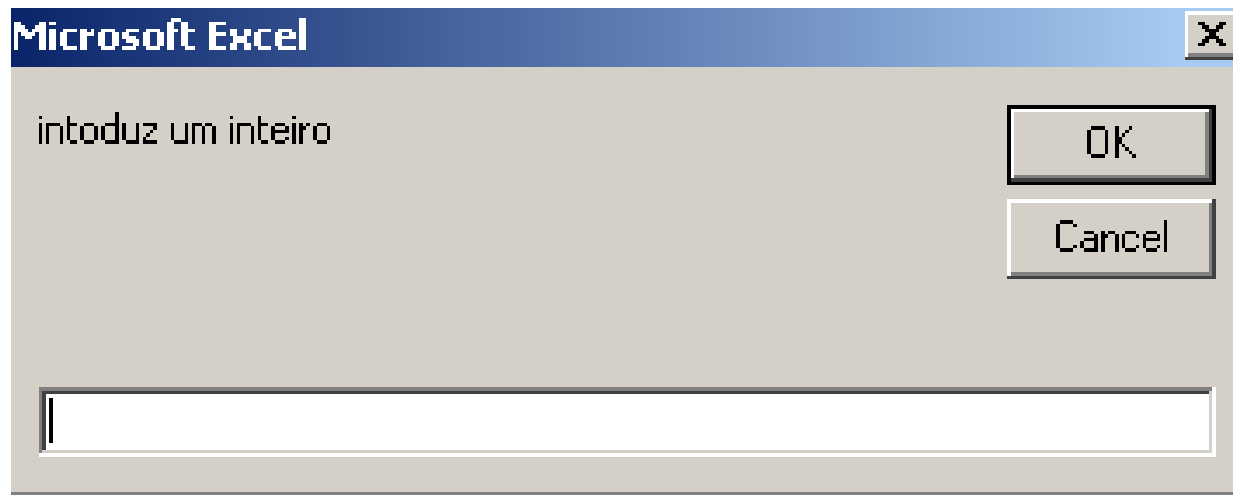
A forma mais simples do Macro interagir com o utilizador é através de MsgBox e de InputBox

- MsgBox → MsgBox("Isto é uma MsgBox!")



Objetos gráficos

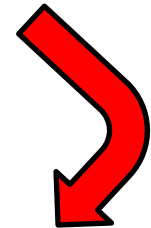
- `InputBox` → `A = InputBox("Intoduz um inteiro")`



Primeiro programa...

(General)

```
Sub PrimeiroPrograma()  
    'Este programa apenas escreve uma mensagem  
    MsgBox ("Aprender a programar é altamente!")  
  
End Sub
```



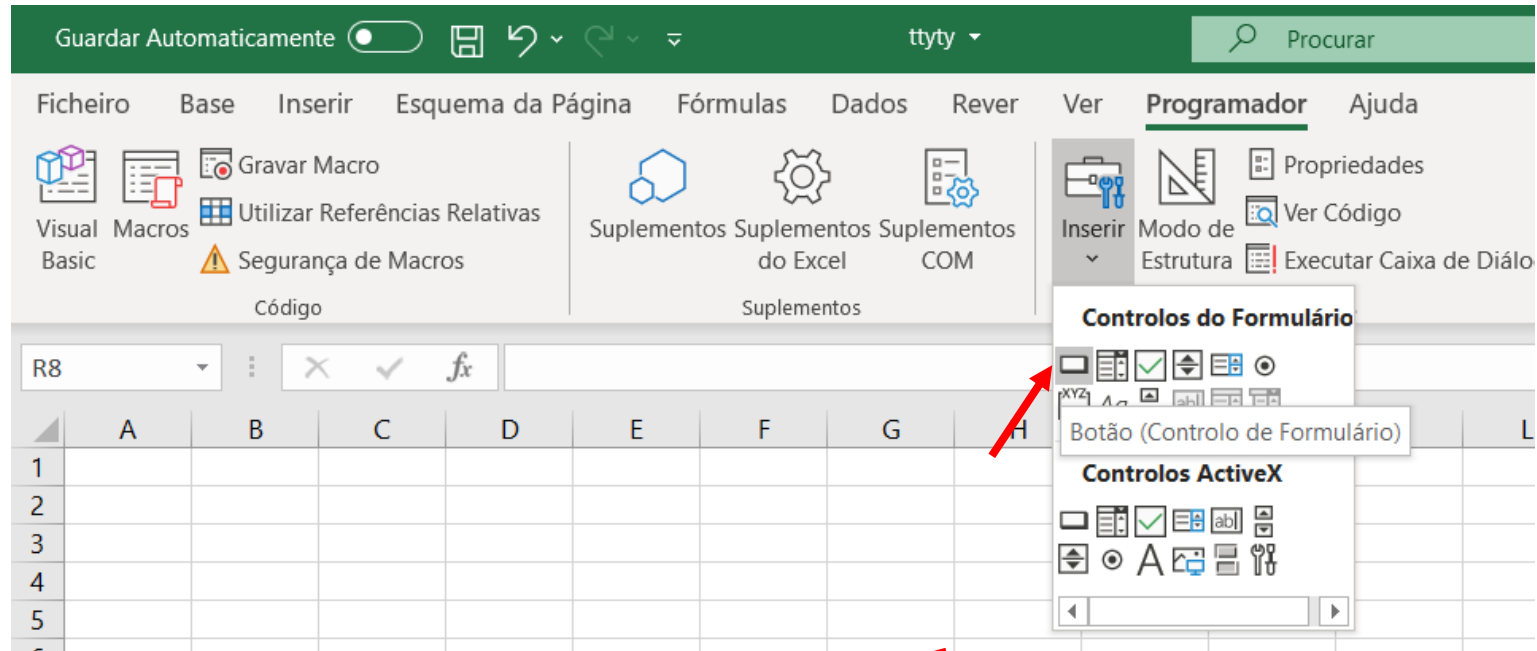
Microsoft Excel



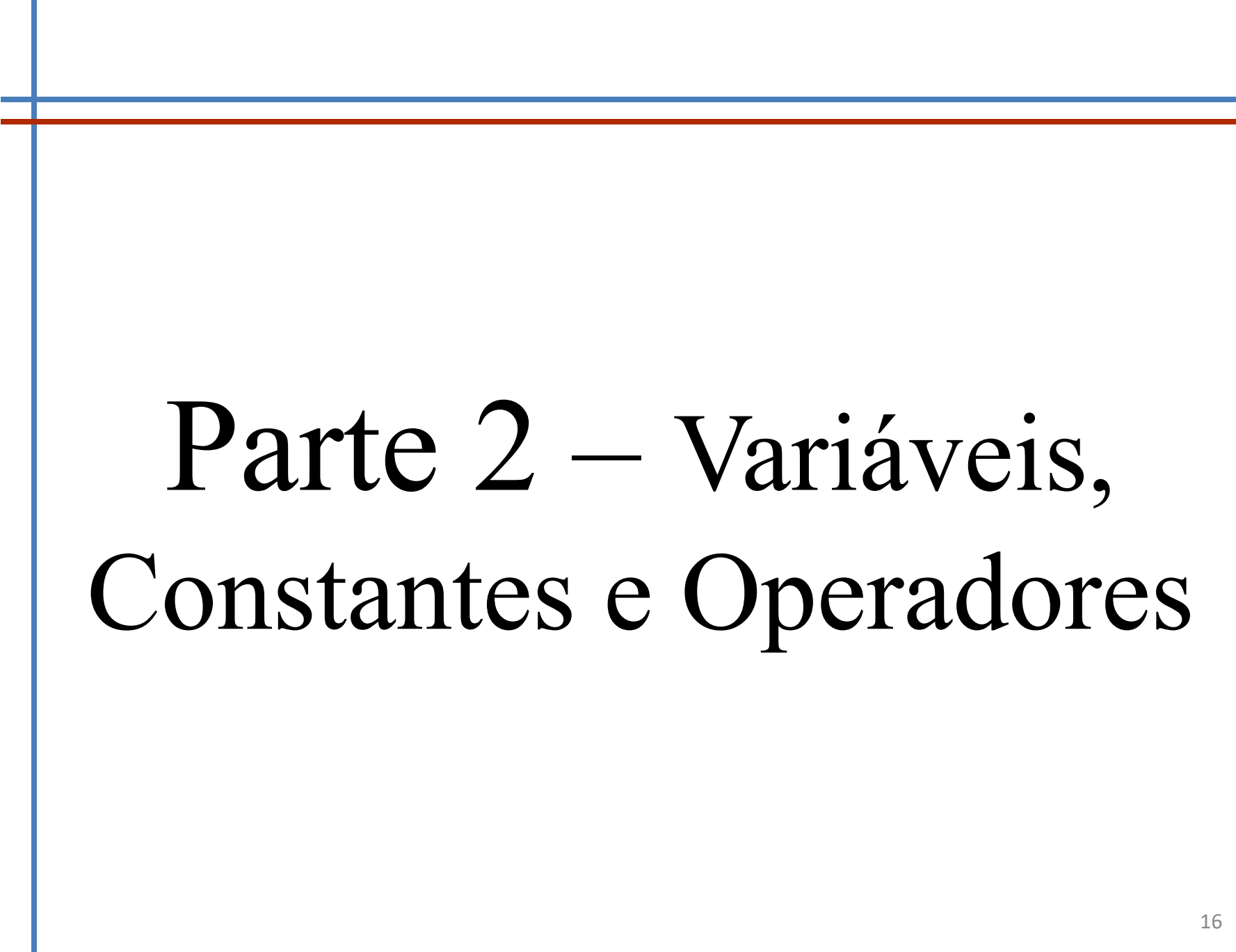
Aprender a programar é altamente!

OK

Associar botão a um macro



Criar macro e
associar ao
botão criado



Parte 2 – Variáveis, Constantes e Operadores

Dicas para escrita de código legível

- Indentação
 - Deve permitir perceber facilmente a estrutura do código, nomeadamente que instruções estão dentro de outras.
- Comentários
 - São as “notas” do programador. Devem ser inseridos para facilitar a compreensão do código.
 - Uma linha que comece por ‘ (plica) ou **Rem** é um comentário que será completamente ignorado pelo compilador.

Variáveis

Uma variável é uma localização de memória em que a informação pode ser guardada de modo a ser usada por um macro.

São caracterizadas por

- Nome
- Tipo de dados que pode armazenar
- Âmbito

Variáveis - declaração

Antes de utilizar uma variável esta deve ser declarada. A declaração de uma variável consiste em criar a variável. Tal declaração é feita atribuindo um nome e um tipo:

Dim nome **As** tipo

Exemplo:

Dim idade **As** Integer

Esta instrução reserva na memória do computador um local para colocar um número inteiro que pode ser acedido através do nome “idade”.

Variável - nome

Regras de atribuição de nomes:

- Deve começar com uma letra ou com *underscore*
- Não pode conter ponto (".");
- Não pode exceder 255 caracteres.
- Deve ser único na área que é usado
- Não pode coincidir com palavras reservadas

Variável - não pode ter os nomes:

And As Boolean ByRef Byte ByVal Call
Case CBool CByte CDate CDbt CInt CLng
Const CSng CStr Date Dim Do Double
Each Else Elseif End EndIf Error False For
Function Get GoTo If Integer Let Lib Long
Loop Me Mid Mod New Next Not Nothing
Option Or Private Public ReDim REM Resume
Select Set Single Static Step String Sub Then
To True Until vbCrLf vbTab With While Xor

Option explicit

- Existe a possibilidade de não declarar explicitamente uma variável. Nesse caso, a primeira vez que o nome é usado a variável é implicitamente declarada. Este procedimento **não é seguro!!**

Devemos então usar a opção **Option Explicit** do VBA. Para tal, devemos:

- Escrever no início do código a instrução: **Option Explicit**
- Developer -> VB -> tool -> options -> require variable declaration

Variável – atribuição de valores

Após declarar uma variável, é possível atribuir-lhe um valor (de acordo com o seu tipo) usando o operador “=” .

Exemplo:

**Criar
Macro**

```
Sub exemplo()  
  Dim idade As Integer  
  idade = 5  
  ...  
  idade = -3  
End Sub
```

**Cria uma variável
com o nome
idade que
assume valores
inteiros**

**Atribuí à variável
o valor “5”**

**Atribuí à variável
o valor “-3”. É
possível?**

Tipos de dados

Numéricos

Byte

Integer

Long

Single

Double

Inteiros

Reais

Outros

String

Boolean

Date

Currency

Variant

Texto

True/False

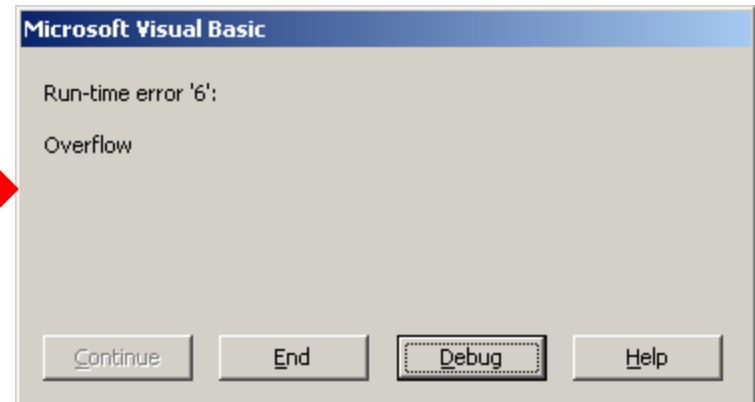
Datas

Geral

Tipo Byte, Integer e Long

	De	Até	Carater para definir
Byte	0	255	
Integer	-32768	32767	%
Long	-2 147 483 648	2 147 483 647	@

```
Sub ErroByte()  
    Dim Valor As Byte  
    Valor=777  
End Sub
```



Tipo Byte, Integer e Long

- Declarar Byte

```
Sub ExemploByte()  
    Dim Idade As Byte  
End Sub
```

- Declarar Integer

```
Sub ExemploInt()  
    Dim Naluno As Integer  
End Sub
```

Ou

```
Sub ExemploInt()  
    Dim Naluno%  
End Sub
```

- Declarar Long

```
Sub ExemploLong()  
    Dim Populacao As Long  
End Sub
```

Ou

```
Sub ExemploLong()  
    Dim Populacao@  
End Sub
```

Tipo Single

Negativos de $-1.401298e^{-45}$ a $-3.402823e^{38}$

Positivos de $1.401298e^{-45}$ a $3.402823e^{38}$

```
Sub ExemploSingle()  
    Dim Distancia As Single  
End Sub
```

```
Sub ExemploSingle()  
    Dim Distancia!  
End Sub
```

Tipo Double

Negativos de $-4.94065645841247e^{-324}$

a $-1.79769313486231e^{308}$

Positivos de $4.94065645841247e^{-324}$

a $1.79769313486231e^{308}$

```
Sub ExemploDouble()  
    Dim Distancia As Double  
End Sub
```

```
Sub ExemploDouble()  
    Dim Distancia#  
End Sub
```

Tipo String

- É um caracter ou um conjunto de caracteres (texto)

```
Sub ExemploString()  
    Dim Nome As String  
End Sub
```

```
Sub ExemploString()  
    Dim Nome$  
End Sub
```

Tipo Currency e Tipo Date

- Entre -922 337 203 685 477.5808 e 922 337 203 685 477.5807
 - O número é escrito normalmente pelo utilizador sendo apresentado no Excel no formato currency.

```
Sub ExemploCurrency()  
    Dim Salario As Currency  
End Sub
```

```
Sub ExemploDate()  
    Dim DataNascimento As Date  
    DataNascimento = #1/10/2010#  
End Sub
```

Tipo Variant

```
Sub ExemploVaR()  
    Dim Nome As Variant  
    Dim Contrato As Variant  
    Dim SalarioHora As Variant  
    Dim InicioContr As Variant  
  
    Nome = "Manuel Joaquim"  
    Contrato = 0  
    SalarioHora = 10  
    InicioContr = #1/1/1980#  
  
End Sub
```

Não é boa prática!
Exige reserva de
muito espaço!!

Âmbito de uma variável

Local do código onde uma variável é reconhecida e pode ser usada. O âmbito de uma variável pode ser local ou global.

```
Option Explicit
```

```
Dim NomeF As String
```

```
Sub ExemploGlobalLocal()
```

```
    Dim NomeP As String
```

```
    NomeP = "Maria"
```

```
    NomeF = "Silva"
```

```
End Sub
```

Variável Global

Acessível em todo o código do módulo

Variável Local

Acessível apenas dentro de ExemploGlobalLocal

Variável - Private/Public

Option Explicit

Private NomeF As String

Public NomeC As String

Sub ExemploPrivPub()

Dim NomeP As String

NomeP = "Maria"

NomeF = "Silva"

NomeC = NomeP & " " & NomeF

End Sub

**Só pode ser acedida
por código do mesmo
módulo**

**Pode ser acedida
por código do
mesmo módulo ou
fora dele**

Variável - Private/Public

Option Explicit

Private NomeC As String

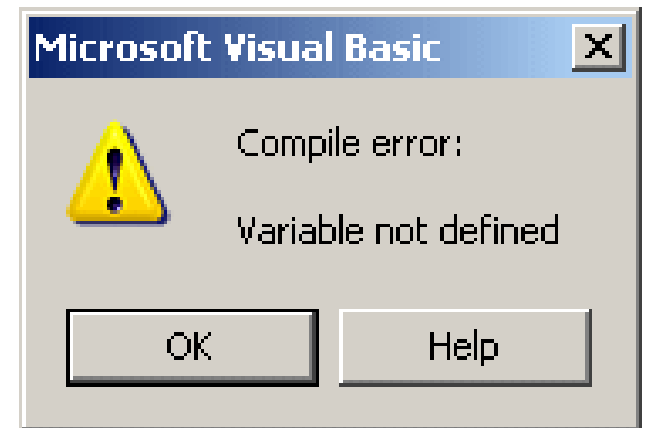
Option Explicit

Private NomeF As String
Private NomeP As String

Sub ExemploPrivPubKO()
 Dim NomeP As String

 NomeP = "Maria"
 NomeF = "Silva"
 NomeC = NomeP & " " & NomeF

End Sub



Constantes

São usadas como “sinónimos” e previnem que um determinado valor seja alterado ao longo do programa.

Tal como as variáveis, também as constantes têm

- um nome alfanumérico
- um tipo de dados

Exemplo:

Const Nome **As** Tipo = expressão

Const IVA **As** Single = 0.23

Operadores e operandos

- Uma operação é uma ação executada com um ou mais valores para modificar um valor ou para obter um novo valor por combinação de valores existentes.

$$y + z$$

- Numa operação existe pelo menos um símbolo e um valor.
- O símbolo é o **operador** +
- O valor ou variável é um **operando** y e z

Operadores e Operandos

- Os operadores podem ser
 - unários $-X$ (envolvem apenas um operando)
 - Binários $X+Y$ (envolvem dois operandos)

Dim (para declarar variáveis) é também um operador

Operadores

- Afectação =
- Continuação de linha _
- Parêntesis ()
- Vírgula ,
- Aspas “ ”
- Dois pontos :
- Concatenação de Strings &
- Mudança de linha **vbCrLf**

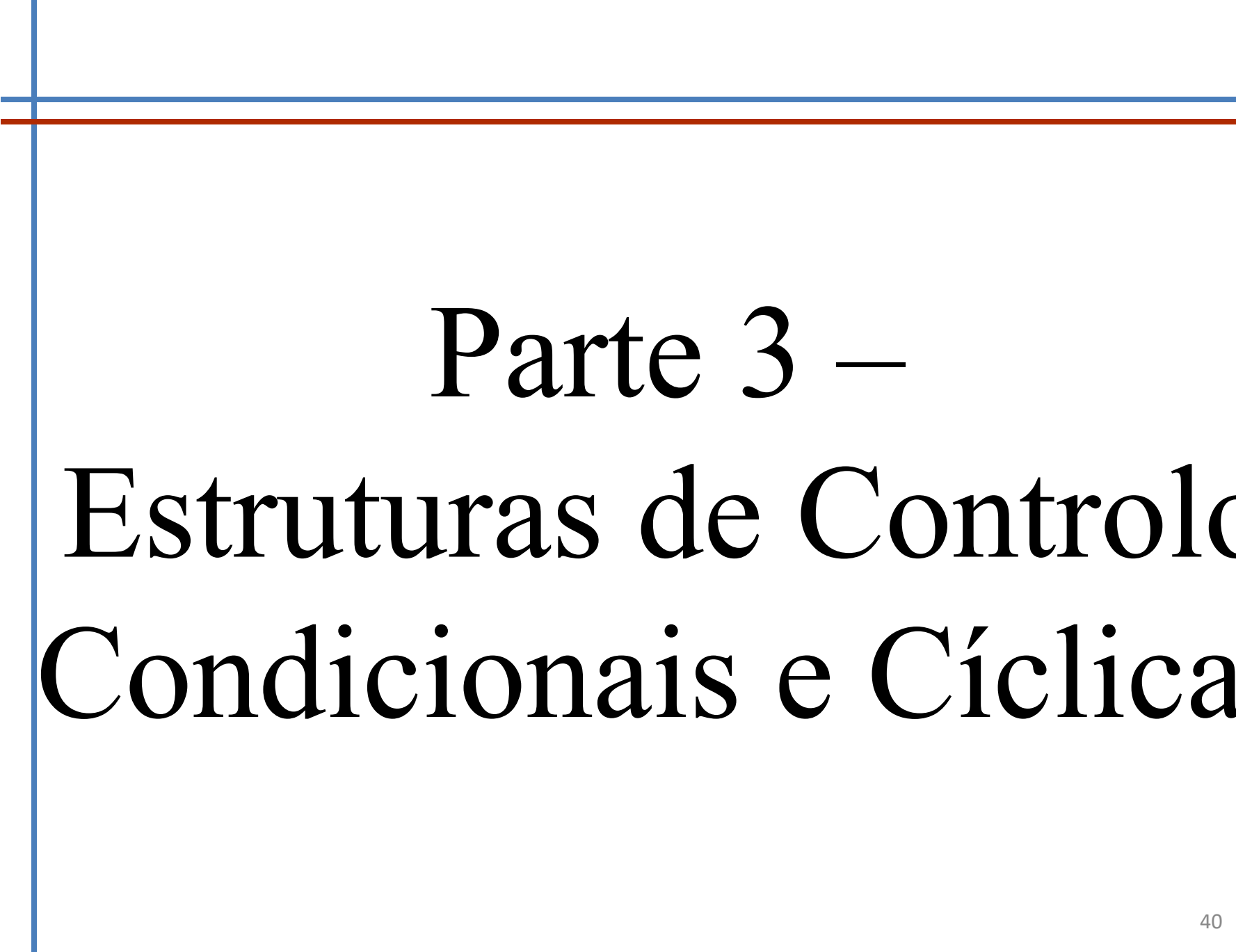
Operadores

Aritméticos

- Adição +
- Subtração −
- Multiplicação *
- Potenciação ^
- Divisão decimal /
- Divisão Inteira \
- Resto da divisão inteira **Mod**

Lógicos

- Menor <
- Maior >
- Menor ou igual <=
- Maior ou igual >=
- Igualdade =
- Desigualdade <>
- Conjunção **And**
- Disjunção **Or**
- Disjunção exclusiva **Xor**
- Negação **Not**



Parte 3 – Estruturas de Controlo Condicionais e Cíclicas

Estruturas de controlo - If...Then...Else

If condição **Then**
instruções

End If

If condição **Then**
instruções

Else
instruções

End If

Estruturas de controlo (encadeadas1)

```
If condição1 Then
    instruções1
Elseif condição2 Then
    instruções2
Elseif condição3 Then
    ...
    instruções n
End If
```

```
If (nota<0) Or (nota>20) Then
    Resultado="Nota Inválida!"
ElseIf (nota<6) Then
    Resultado="Mau"
ElseIf (nota<10) Then
    Resultado="Mediocre"
ElseIf (nota<14) Then
    Resultado="Suficiente"
ElseIf (nota<17) Then
    Resultado="Bom"
Else
    Resultado="Muito Bom"
End If
```

Estruturas de controlo (encadeadas2)

```
If condição1 Then
    instruções1
Else
    If condição2 Then
        instruções2
    Else
        If condição3 Then
            instruções3
        Else
            instruções4
        End If
    End If
End If
```

Estruturas de controlo - selection ...case

Select case expressão

Case instância
instrução1

Case condição
instrução2

Case Range Is
instrução3

Case Else
instrução4

End Select

```
Select Case nota
```

```
Case 0 To 6
```

```
    resultado = "Mau"
```

```
Case 7 To 9
```

```
    resultado = "Mediocre"
```

```
Case 10 To 13
```

```
    resultado = "Suficiente"
```

```
Case 14 To 16
```

```
    resultado = "Bom"
```

```
Case 17 To 20
```

```
    resultado = "Muito Bom"
```

```
End Select
```

```
Select Case nota
```

```
Case Is > 10
```

```
    resultado = "Aprovado"
```

```
Case Is < 10
```

```
    resultado = "Reprovado"
```

```
Case Else
```

```
    resultado = "Tangente!"
```

```
End Select
```

Estruturas de Controlo

Do While condição
instruções

Loop

```
Contador=1
Soma=0
Do While contador <= n
    Soma=Soma+contador
    contador=contador+1
Loop
```

Do
instruções

Loop While condição

```
Contador=n
Fact=1
Do
    Fact=Fact*contador
    contador=contador-1
Loop While contador > 1
```

Estruturas de Controlo

Do Until condição
instruções

Loop

```
contador = 1
Soma = 0
Do Until contador > n
    Soma = Soma + contador
    contador = contador + 1
Loop
```

Do
instruções

Loop Until condição

```
Contador=0
Fact=1
Do
    contador = contador + 1
    Fact = Fact * contador
Loop Until contador >= n
```

Estruturas de Controlo

For contador= v_inicial **To** v_final

instruções

Next contador

```
For i = 1 To n Step 2
    Soma = Soma + i
Next i
```

For contador= v_inicial **To** v_final **Step** passo

instruções

Next contador

For Each objecto numa colecção
instruções

Next

Estruturas de Controlo (encadeadas)

```
For contador1= v_inicial1 To v_final1  
    For contador2= v_inicial2 To v_final2  
        instruções  
    Next contador2  
Next contador1
```

```
Dim i As Integer  
Dim j As Integer  
For i = 1 To n  
    For j = 1 To n  
        Worksheets("Sheet1").Cells(i, j).Value = i + j  
    Next j  
Next i
```

Instrução exit

As instruções **Exit For** and **Exit Do** permitem interromper um ciclo for e um ciclo while, respetivamente.

No caso de dois ciclos encadeados, a instrução **exit** interrompe apenas um deles (dependendo da posição onde é colocada).

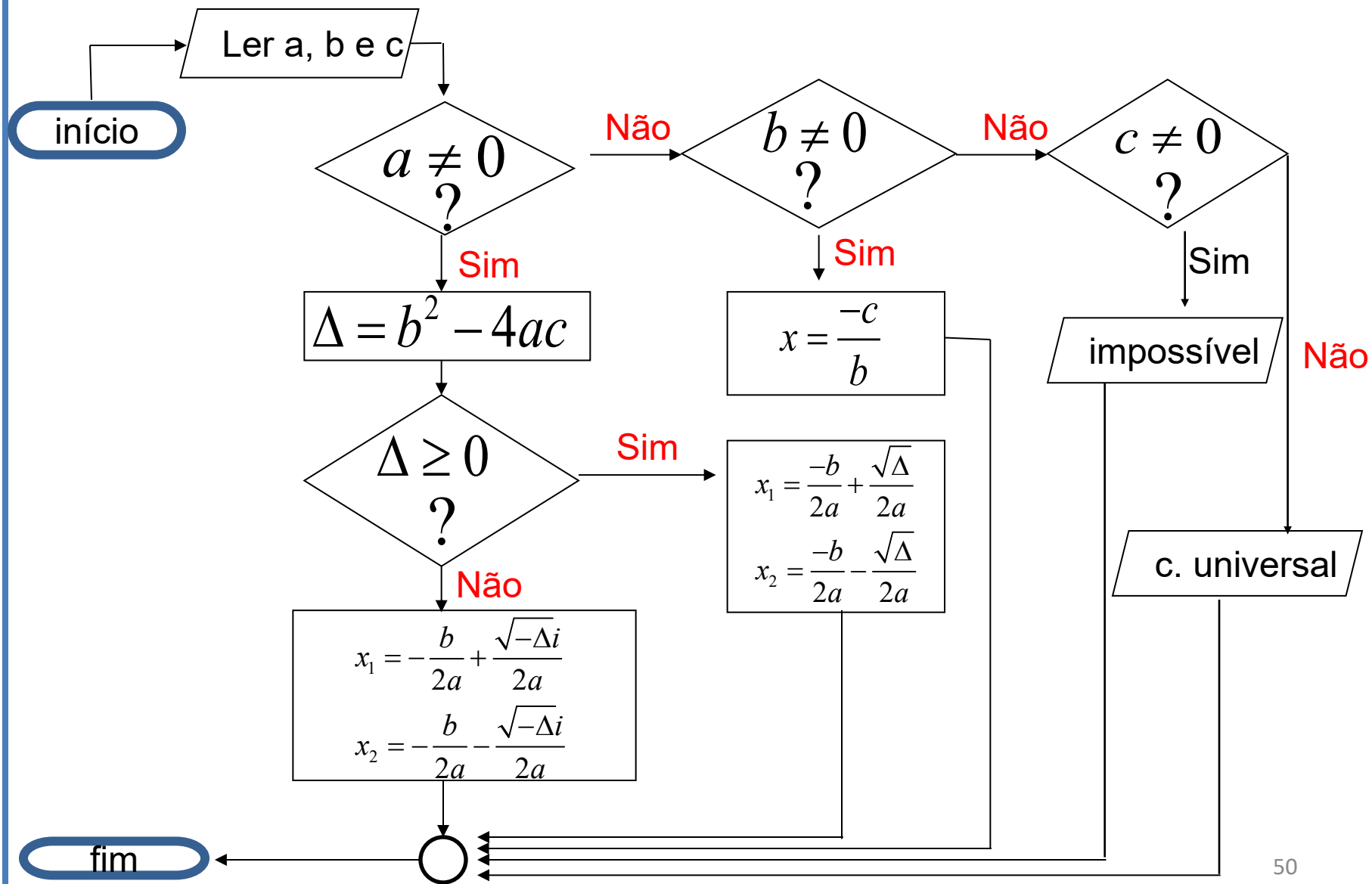
```
For i = 1 To 3
  For j = 1 To 10
    MsgBox ( i & " " & j)
    If j=2 Then
      Exit For
    End If
  Next j
Next i
```

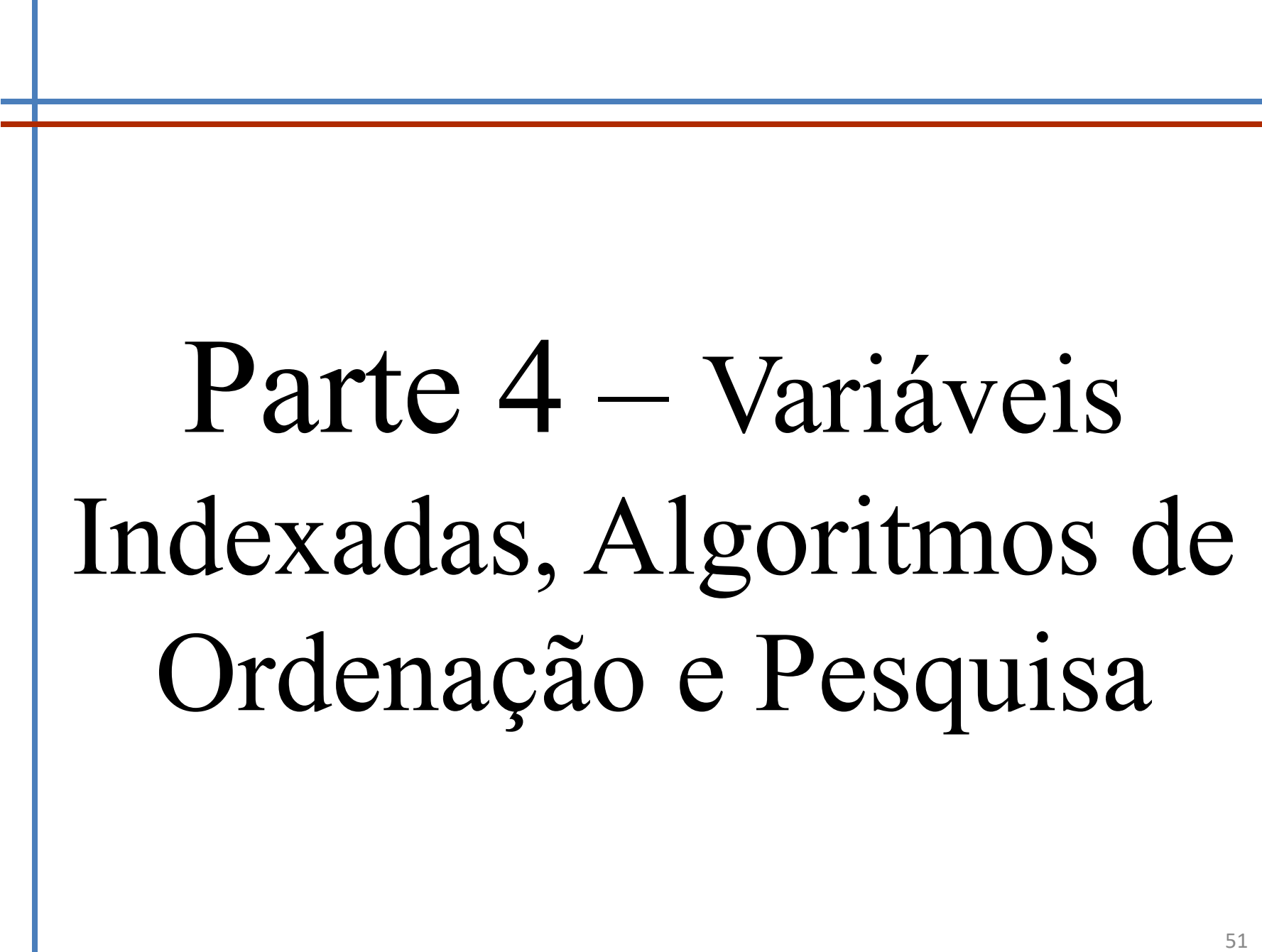


```
1 1
1 2
2 1
2 2
3 1
3 2
```

Resolver Equação

$$ax^2 + bx + c = 0$$





Parte 4 — Variáveis Indexadas, Algoritmos de Ordenação e Pesquisa

Variáveis indexadas

- **Array** (vector)

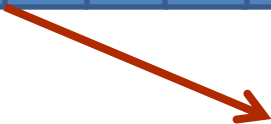
Permitem armazenar numa variável vários valores desde que sejam todos do mesmo tipo

Variável b



12	15	18	18	12	15	16	10	13	14
----	----	----	----	----	----	----	----	----	----

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---



$b(0)=12$

$b(1)=15$

...

$b(9)=14$

Valor do índice deve ser inteiro e por omissão começa em zero

Variáveis indexadas

Exemplo com strings

`nomes(0)="Pedro"`

`nomes(1)="Mariana"`

...

`nomes(6)="Manuel"`

Variável *nomes*



Pedro	Mariana	Joana	António	Guilherme	Maria	Manuel
0	1	2	3	4	5	6

Variáveis indexadas

- **Declarações Estáticas**

Dim nome(maior_índice) **As** Tipo

Dim nome(menor_índice **To** maior_índice) **As** Tipo

Exemplos:

Dim alunos(99) **As** String (de 0 a 99)

Dim Medias(12) **As** Double (de 0 a 12)

Dim alunosNovos(100 **To** 200) **As** String (de 100 a 200)

Dim Medias(-100 **To** 100) **As** integer (de -100 a 100)

Adicionalmente, com **Dim** os valores numéricos são inicializados a zeros e os alfanuméricos a strings nulas

Variáveis indexadas

- **Declaração Dinâmica**

Quando não sabemos a dimensão do vector no momento da sua declaração

Dim nome_do_vector() **As** Tipo

...

ReDim nome_do_vector(1 **To** n)

...

ReDim Preserve nome_do_vector(1 **To** n+k)

Quanto ainda
não sabemos a
dimensão

Quanto já
sabemos a
dimensão

Se for necessário ajustar a dimensão sem perder os valores já existentes no vetor. (Apenas é possível alterar o limite superior)

Variáveis indexadas

- **Utilidade**

Exemplo: uma empresa de produtos lacteos tem 100 produtos cada um com o seu preço e pretende atualizar o IVA que era de 6% e passou a ser de 23%.

100 variáveis

Declaração

Dim preço1 **As** Single

Dim preço2 **As** Single

...

Dim preço100 **As** Single

Dim atualiza **As** Single

atualiza = 0.23/0.06

Atualização

preço1 = atualiza*preço1

preço2 = atualiza*preço2

...

preço100 = atualiza*preço100

1 variável indexada com 100 elementos

Declaração

Dim preços(1 To 100) **As** Single

Dim atualiza **As** Single

Dim i **As** Integer

atualiza = 0.23/0.06

Atualização

For i=1 **To** 100

preços(i) = atualiza*preços(i)

Next i

Variáveis indexadas - Exemplo

Sub Somatorio()

Dim dados(1 To 10) **As** Integer

Dim i **As** Integer

Dim Soma **As** Integer

Soma = 0

For i = 1 **To** 10

dados(i) = ActiveSheet.Cells(i, 1)

Soma = Soma + dados(i)

Next i

MsgBox ("Soma=" & Soma)

ReDim Preserve dados(1 **To** 20)

For i = 11 **To** 20

dados(i) = ActiveSheet.Cells(i, 1)

Soma = Soma + dados(i)

Next i

MsgBox ("Soma=" & Soma)

End Sub

← Lê o valor que está na célula (i,1) da folha do Excel

← Mantêm os 10 valores do vetor e cria lugar para mais 10

Instruções LBound e UBound

Indicam qual a primeira e a última posição do vetor.

Sub exemplo()

Dim v(2 **To** 5) **As** Integer

Dim s **As** String

Dim i **As** Integer

v(2)=10

v(3)=20

v(4)=30

v(5)=40

For i = LBound(v) **To** UBound(v)

s= s & " " & v(i)

Next i

MsgBox ("O vetor tem " & UBound(v)-LBound(v)+1 & "elementos: " & s)

End Sub

Algoritmos de pesquisa

- Dado um vector v determinar a (1ª) ocorrência (chave/posição) de um determinado elemento.

$V(i)$	12	15	18	18	12	15	16	10	13	14
i	0	1	2	3	4	5	6	7	8	9

Pesquisa Sequencial

Dado um elemento x , percorrer de forma sequencial o vetor v , comparando cada x com $v(i)$ até encontrar x .

```
Posicao=-1
For i = 1 To n
    If v(i)=x Then
        Posicao=i
        Exit For
    End If
Next i

If Posicao = -1 Then
    MsgBox("Não existe no vetor")
Else
    MsgBox("Existe na posição " & Posicao)
End If
```

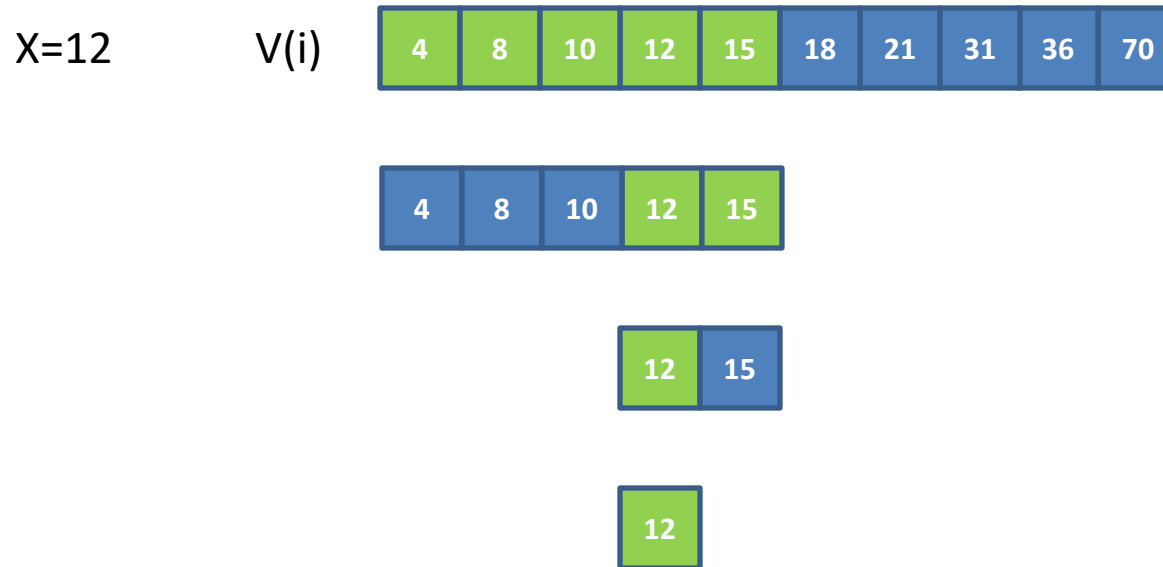
```
For i = 1 To n
    If v(i)=x Then
        Exit For
    End If
Next i

If i = n+1 Then
    MsgBox("Não existe no vetor")
Else
    MsgBox("Existe na posição " & i )
End If
```

Complexidade $O(n)$

Pesquisa Binária

- Assume que o vector contém apenas dados numéricos e encontra-se ordenado.
- Divide sucessivamente o vector ao meio até que x seja encontrado.



Pesquisa Binária

Esquerda = 0

Direita = n-1

Posicao=-1

Do While Esquerda<=Direita **And** Posicao=-1

Meio = (Esquerda + Direita)\2

If v(Meio)=x **Then**

Posicao=Meio

Exit Do

Elseif v(Meio)> x **Then**

Direita=Meio-1

Else

Esquerda=Meio +1

End If

Loop

Complexidade $O(\log(n))$

Algoritmos de Ordenação

Porquê ordenar vetores?

- Rapidez na pesquisa
- Agrupamento dos dados em classes
- Calculo de estatísticas descritivas: max, min, mediana, etc.
- Algoritmos mais complexos usam algoritmos de ordenação

Algoritmos de Ordenação

É muito importante ter em conta a complexidade do algoritmo.

Métodos de ordenação (mais comuns):

- Seleção (Selection Sort)
- Inserção (Insertion Sort)
- Troca (Bubble Sort, Quicksort)

Selection Sort

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	6	2	6
---	---	---	---	---	---	---

1	3	4	4	6	2	6
---	---	---	---	---	---	---

1	3	4	4	6	2	6
---	---	---	---	---	---	---

1	2	4	4	6	3	6
---	---	---	---	---	---	---

1	2	4	4	6	3	6
---	---	---	---	---	---	---

1	2	3	4	6	4	6
---	---	---	---	---	---	---

1	2	3	4	6	4	6
---	---	---	---	---	---	---

1	2	3	4	6	4	6
---	---	---	---	---	---	---

1	2	3	4	6	4	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

**Seleciona o elemento mais
pequeno e coloca-o no
início.**

Insertion Sort

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	6	2	6
---	---	---	---	---	---	---

3	4	4	1	6	2	6
---	---	---	---	---	---	---

3	4	4	1	6	2	6
---	---	---	---	---	---	---

3	4	4	1	6	2	6
---	---	---	---	---	---	---

3	4	4	1	6	2	6
---	---	---	---	---	---	---

1	3	4	4	6	2	6
---	---	---	---	---	---	---

1	3	4	4	6	2	6
---	---	---	---	---	---	---

1	3	4	4	6	2	6
---	---	---	---	---	---	---

1	3	4	4	6	2	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

Seleciona os elementos por ordem e vai-os colocando na posição correta.

Bubble Sort

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	6	2	6
---	---	---	---	---	---	---

4	3	4	1	2	6	6
---	---	---	---	---	---	---

4	3	4	1	2	6	6
---	---	---	---	---	---	---

4	3	1	4	2	6	6
---	---	---	---	---	---	---

4	1	3	4	2	6	6
---	---	---	---	---	---	---

1	4	3	4	2	6	6
---	---	---	---	---	---	---

1	4	3	4	2	6	6
---	---	---	---	---	---	---

1	4	3	4	2	6	6
---	---	---	---	---	---	---

1	4	3	4	2	6	6
---	---	---	---	---	---	---

1	4	3	2	4	6	6
---	---	---	---	---	---	---

1	4	2	3	4	6	6
---	---	---	---	---	---	---

1	2	4	3	4	6	6
---	---	---	---	---	---	---

1	2	4	3	4	6	6
---	---	---	---	---	---	---

1	2	4	3	4	6	6
---	---	---	---	---	---	---

1	2	4	3	4	6	6
---	---	---	---	---	---	---

1	2	4	3	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

1	2	3	4	4	6	6
---	---	---	---	---	---	---

Exemplo Bubble Sort

Procedimento Bubble(vector de inteiros $A[1...n]$)

//para ordenar por ordem crescente A

Para $i=1$ **até** $n-1$ (1)

Para $j=i+1$ **até** n (2)

Se $A(i) > A(j)$ **então** (3)

 'trocar $A(i)$ com $A(j)$

$\text{temp} = A(i)$ (4)

$A(i) = A(j)$ (5)

$A(j) = \text{temp}$ (6)

Complexidade

Análise do pior caso:

7	6	5	4	3	2	1
---	---	---	---	---	---	---

- Os três algoritmos descritos têm complexidade $O(n^2)$.
- Existem algoritmos mais eficientes com complexidade $O(n \log(n))$. Exemplos: Mergesort, Heapsort, etc..
- No entanto: Selection sort, Insertion sort e Bubble sort são “rápidos” para vetores de pequena dimensão e fáceis de implementar.

Variáveis indexadas multidimensionais

Matrizes

Dim nome_matriz(num_linhas,num_colunas) **As** Tipo

Exemplo:

Dim Mat1(1 To 4,1 To 3) **As** Integer →

0	0	0
0	0	0
0	0	0
0	0	0

Mat1(3,3)=6

Mat1(1,1)=1

...

Mat1(4,3)=13



1	5	7
3	2	4
11	10	6
8	12	13

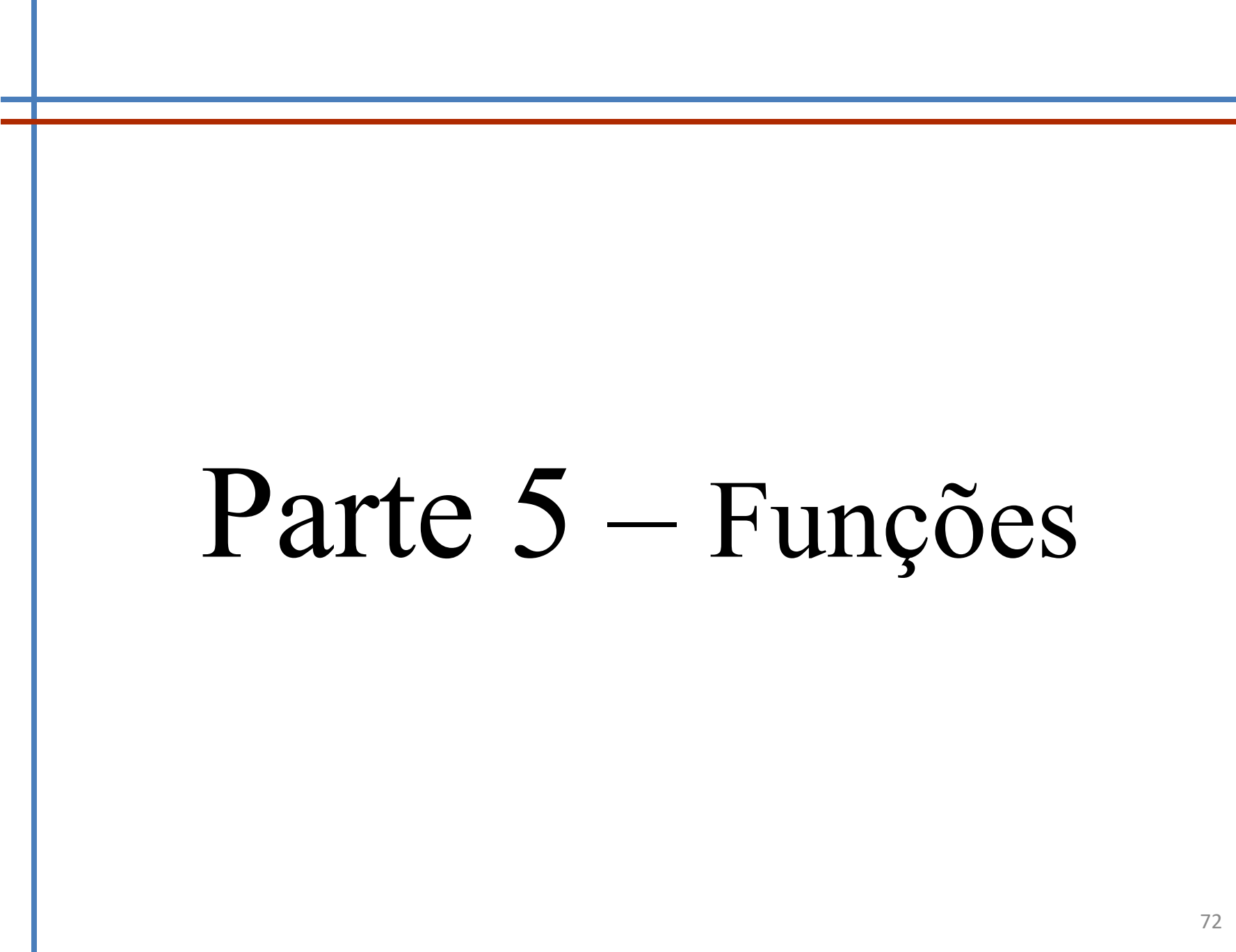
Variáveis indexadas

- **Manipulação de Matriz ciclos encadeados**

```
Public Sub mat()  
    Dim mat(1 To 4, 1 To 4) As Integer  
    Dim i As Integer  
    Dim j As Integer  
    Dim k As Integer  
    For i = 1 To 4  
        For j = 1 To 4  
            mat(i, j) = Cells(i, j)  
        Next j  
    Next i  
    k = 6  
    For i = 1 To 4  
        For j = 1 To 4  
            Cells(i + k, j + k) = mat(i, j)  
        Next j  
    Next i  
End Sub
```

Lê uma matriz 4x4 a partir de uma folha de Excel e reescreve-a num lugar diferente da folha do Excel.





Parte 5 — Funções

Procedimentos e Funções

- Os **procedimentos** ou subrotinas destinam-se a realizar um conjunto de tarefas mas não têm necessariamente que devolver qualquer resultado
- Uma **função** destina-se a realizar um conjunto de tarefas e a devolver um resultado

Funções

Function Nome (arg1,arg2,...) Tipo de dado

‘ Lista de instruções

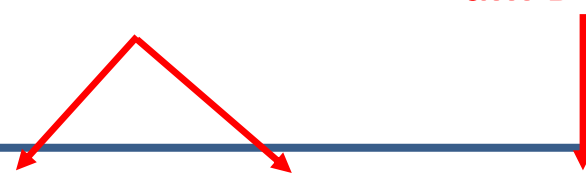
Nome=resultado

End Function

Exemplo:

Recebe dois
Doubles

Devolve
um Double



```
Function Area(a As Double, c As Double) As Double
```

```
    Area= a*c
```

```
End Function
```

Exemplo: Função fatorial

```
Private Sub Combina()  
    Dim n As Integer  
    Dim p As Integer  
    Dim Comb As Double  
    n = InputBox("Numero de elementos")  
    p = InputBox("Grupos de quantos elementos")  
    If n > 0 And p > 0 And n >= p Then  
        Comb = fatorial(n) / (fatorial(n - p) * fatorial(p))  
        MsgBox (" num. de combinações " & Comb)  
    Else  
        MsgBox (" Erro, dados inválidos ")  
    End If  
End Sub
```

**Função
Auxiliar**



```
Public Function fatorial(n As Integer) As Double  
    Dim i As Integer, f As Double  
    f = 1  
    For i = 2 To n  
        f = f * i  
    Next i  
    fatorial = f  
End Function
```

**Programa
principal**



Procedimentos

Sub Nome (arg1,arg2,...)

‘ Lista de instruções

End Sub

Call Nome()

```
Public Sub exemplo()  
    Dim x As Integer  
    Dim y As Integer  
    x = InputBox("introduza um inteiro")  
    y = InputBox("introduza um inteiro")  
    MsgBox "(" & x & "," & y & ")=>" & " Soma:" & x + y  
End Sub
```

```
Public Sub chamada()  
    Call exemplo  
End Sub
```

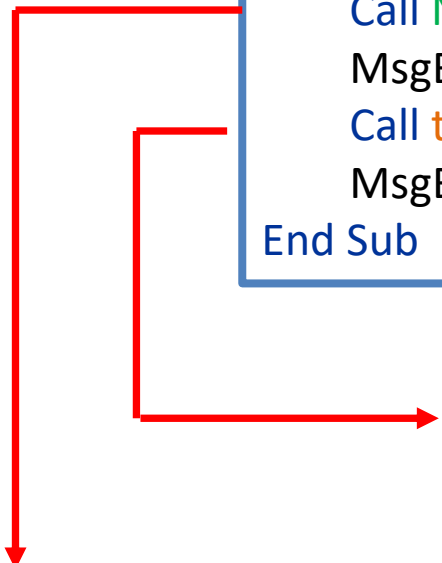
*A instrução **Exit Sub** permite terminar imediatamente um procedimento.

Passagem de parâmetros

- **ByVal** O que é passado à função ou ao procedimento é uma cópia do valor corrente da variável.
Dentro da função ou procedimento esse valor é alterados mas no retorno ao programa o valor da variável continua igual ao que era na altura da chamada
- **ByRef** O que é passado à função ou ao procedimento é o endereço da variável, uma autorização para alterar o seu valor. Dentro da função ou procedimento o valor da variável é alterado e no retorno ao programa essa variável ira assumir um novo valor (de acordo com as alterações feitas)

Exemplo de passagem de argumentos

```
Public Sub principal()  
    Dim x As Integer  
    Dim y As Integer  
    x = 1:y = 2  
    MsgBox ("antes de nãoTroca (x,y)=(" & x & "," & y & ")")  
    Call NãoTroca(x, y)  
    MsgBox ("depois de nãoTroca (x,y)=(" & x & "," & y & ")")  
    Call troca(x, y)  
    MsgBox ("depois de Troca (x,y)=(" & x & "," & y & ")")  
End Sub
```



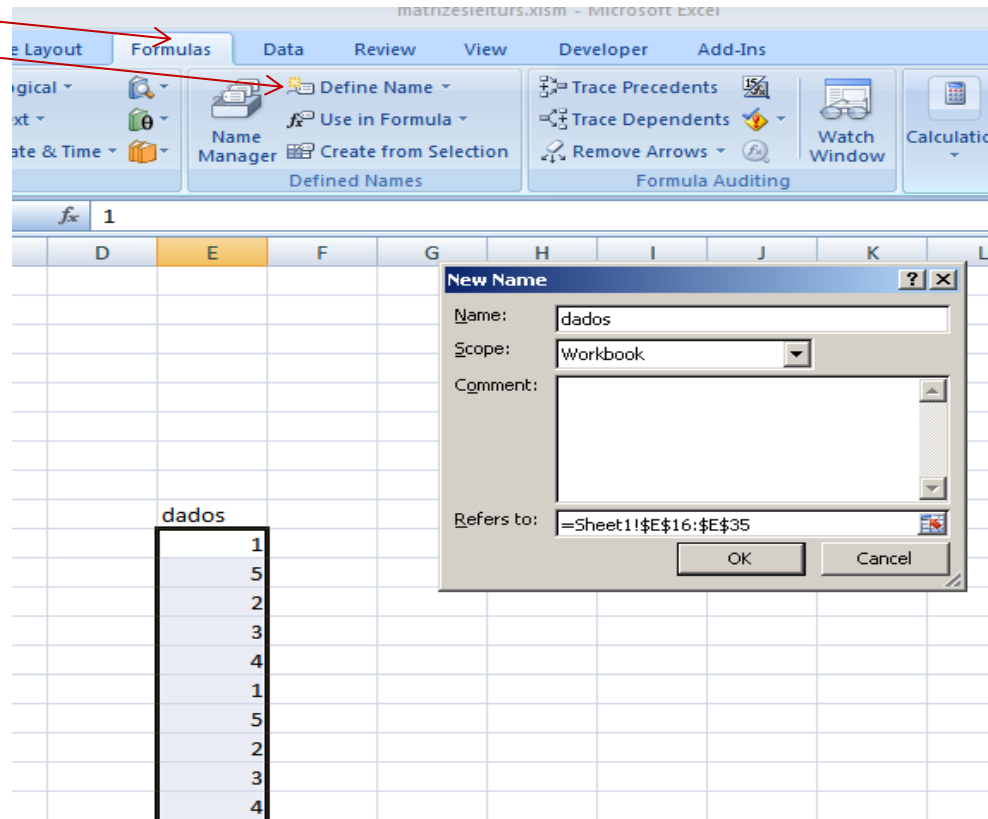
The diagram illustrates the flow of control between three subroutines. A red arrow originates from the 'Call NãoTroca(x, y)' line in the 'principal()' subroutine and points down to the 'NãoTroca' subroutine. Another red arrow originates from the 'Call troca(x, y)' line in the 'principal()' subroutine and points down to the 'troca' subroutine.

```
Public Sub troca(ByRef a As Integer, ByRef b As Integer)  
    Dim temp As Integer  
    temp = a : a = b : b = temp  
End Sub
```

```
Public Sub NãoTroca(ByVal a As Integer, ByVal b As Integer)  
    Dim temp As Integer  
    temp = a : a = b : b = temp  
End Sub
```

Usar as funções do excel

Definir nome => dados



```
Med = Application.worksheetFunction.Average(Range("dados"))
```

Recursividade

Uma função recursiva é uma função que se chama a ela própria. Neste tipo de função é extremamente importante definir cuidadosamente o critério de paragem.

Exemplos

- Fatorial
- Números de Fibonacci
- Algoritmo de Euclides
- Torres de Hanoi (dupla)

Exemplo: Fatorial

fatorial(6)
6 * fatorial(5)
6 * 5 * fatorial(4)
6 * 5 * 4 * fatorial(3)
6 * 5 * 4 * 3 * fatorial(2)
6 * 5 * 4 * 3 * 2 * fatorial(1)
6 * 5 * 4 * 3 * 2 * 1

6 * 5 * 4 * 3 * 2
6 * 5 * 4 * 6
6 * 5 * 24
6 * 120
720

```
Function fatorial(n As Integer) As Integer
    If n = 1 Then
        fatorial = 1
    Else
        fatorial = n * fatorial(n - 1)
    End If
End Function
```

Exemplo: Algoritmo de Euclides

Dados 2 números inteiros m e n ,
calcular o seu máximo divisor
comum $\text{mdc}(m,n)$.

Ler m e n (inteiros diferentes de 0);

Enquanto $(n \neq 0)$

Torne $\text{resto} = m \bmod n$

$m = n$

$n = \text{resto}$

Escrever $\text{mdc}(m,n) = m$; **STOP**

Function $\text{mdc}(a, b)$

If $b = 0$ Then

$\text{mdc} = a$

Else

$\text{mdc} = \text{mdc}(b, a \bmod b)$

End If

End Function

Exemplo: Fibonacci

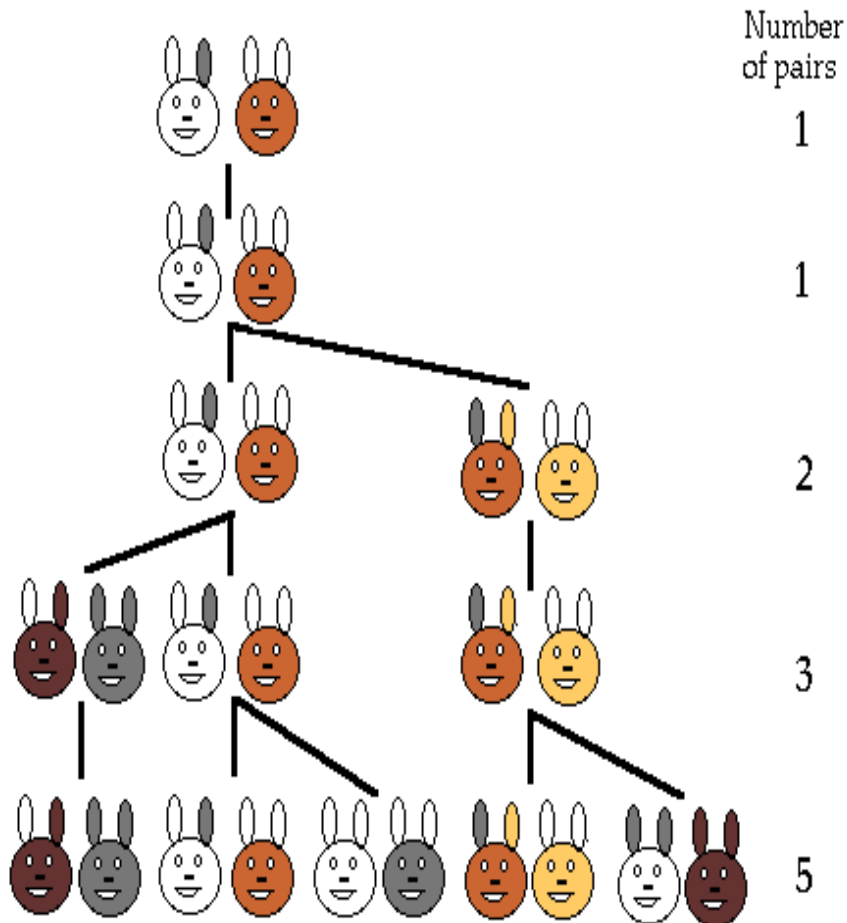
- Mês 0 No início da experiência existe apenas um par de coelhos.
- Mês 1 Após um mês, os coelhos acasalaram mas ainda não deram à luz (portanto existe somente um par de coelhos).
- Mês 2 Neste mês já a fêmea deu à luz um par de coelhos. Existem agora dois pares de coelhos.
- Mês 3 Depois de 3 meses, o par inicial de coelhos dá à luz mais um par de coelhos. No entanto, o segundo par acasala. Isto faz então um total de três pares.
- Mês 4 Aos 4 meses, o par original tem mais um par de coelhos. O par nascido no mês 2 também dá à luz. O par de coelhos nascido no mês 3 acasalam, mas ainda não dão à luz. Isto faz um total de cinco pares.
- Mês 5 Aos 5 meses, todos os pares que nasceram até há dois meses dão à luz. Isto totaliza oito pares.

$$\text{fib}(0) = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2), \text{ para } n > 1$$

Exemplo: Fibonacci



'calcula o n-ésimo número de Fibonacci

Function Fib(n)

If n = 0 Or n = 1 Then

Fib = 1

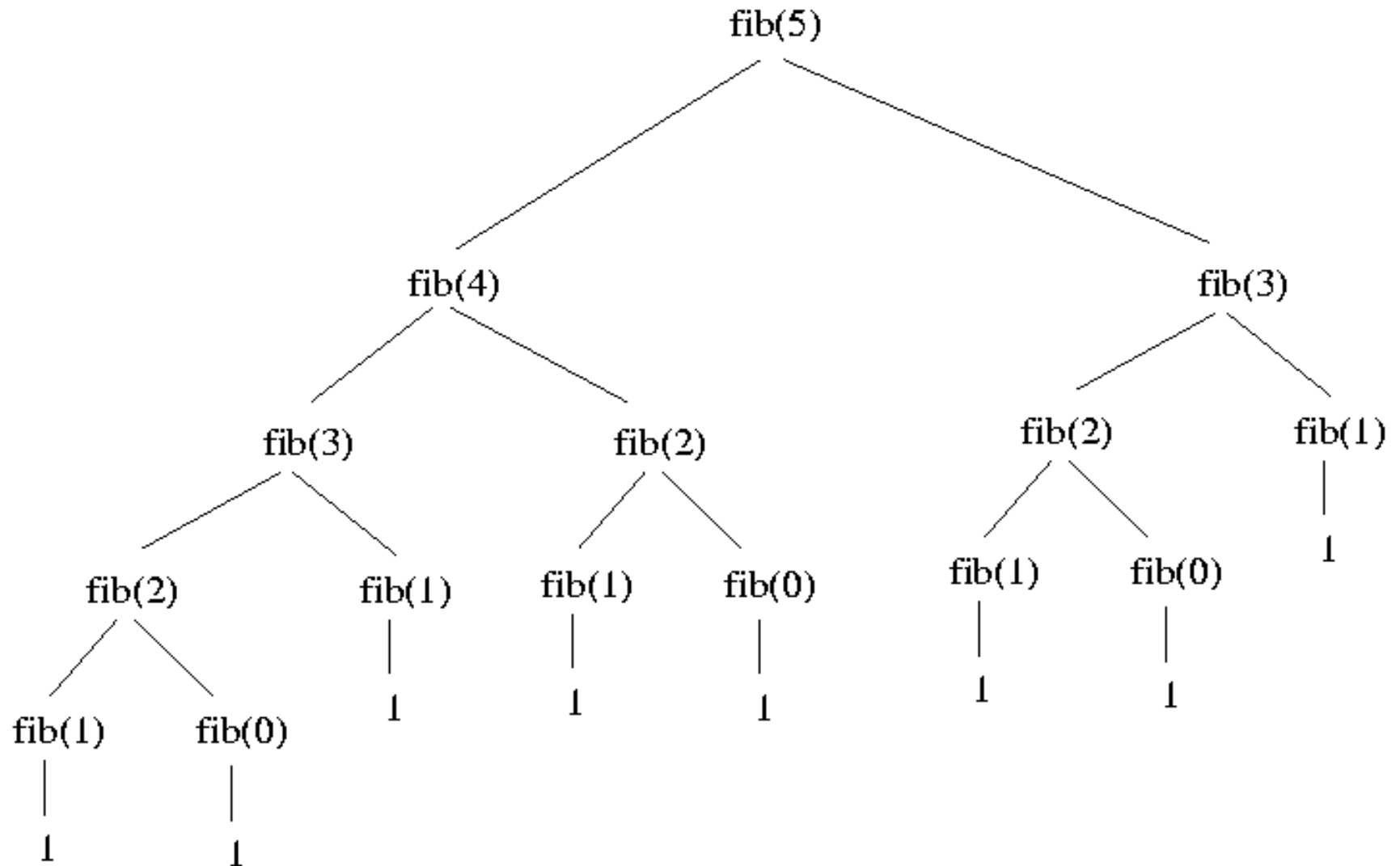
Else

Fib = Fib(n - 1) + Fib(n - 2)

End If

End Function

Exemplo: Fibonacci



Exemplo: Torres de Hanoi

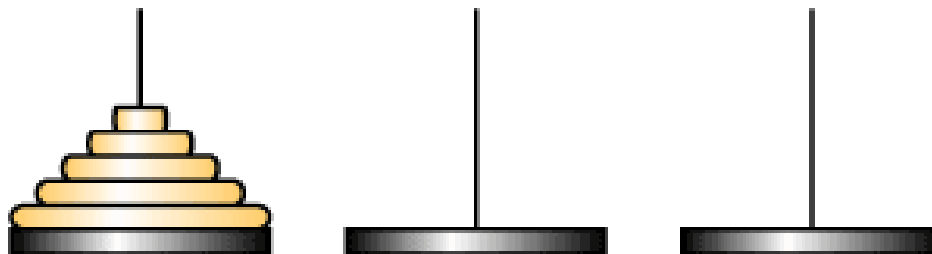
O problema das Torres de Hanói foi inicialmente proposto pelo matemático francês Edouard Lucas, em 1883.

“No grande templo de Brahma em Benares, numa bandeja de metal sob a cúpula que marca o centro do mundo, três agulhas de diamante servem de pilar a sessenta e quatro discos de ouro puro. Incansavelmente, os sacerdotes transferem os discos, um de cada vez, de agulha para agulha, obedecendo sempre à lei imutável de Brahma: Nenhum disco se poderá sobrepor a um menor. No início do mundo todos os sessenta e quatro discos de ouro, foram dispostos na primeira das três agulhas, constituindo a Torre de Brahma. No momento em que o menor dos discos for colocado de tal modo que se forme uma vez mais a Torre de Brahma numa agulha diferente da inicial, tanto a torre como o templo serão transformados em pó e o ribombar de um trovão assinalará o fim do mundo.”

Exemplo: Torres de Hanoi

Restrições a obedecer na movimentação dos discos

1. apenas se pode mover um único disco por vez;
2. só se podem mover discos que estão no topo;
3. nenhum disco pode ser colocado sobre outro menor;



Para mover o disco maior para o poste destino

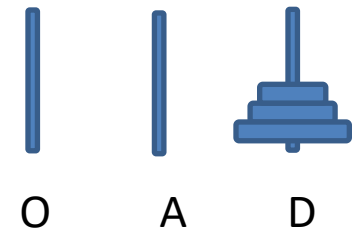
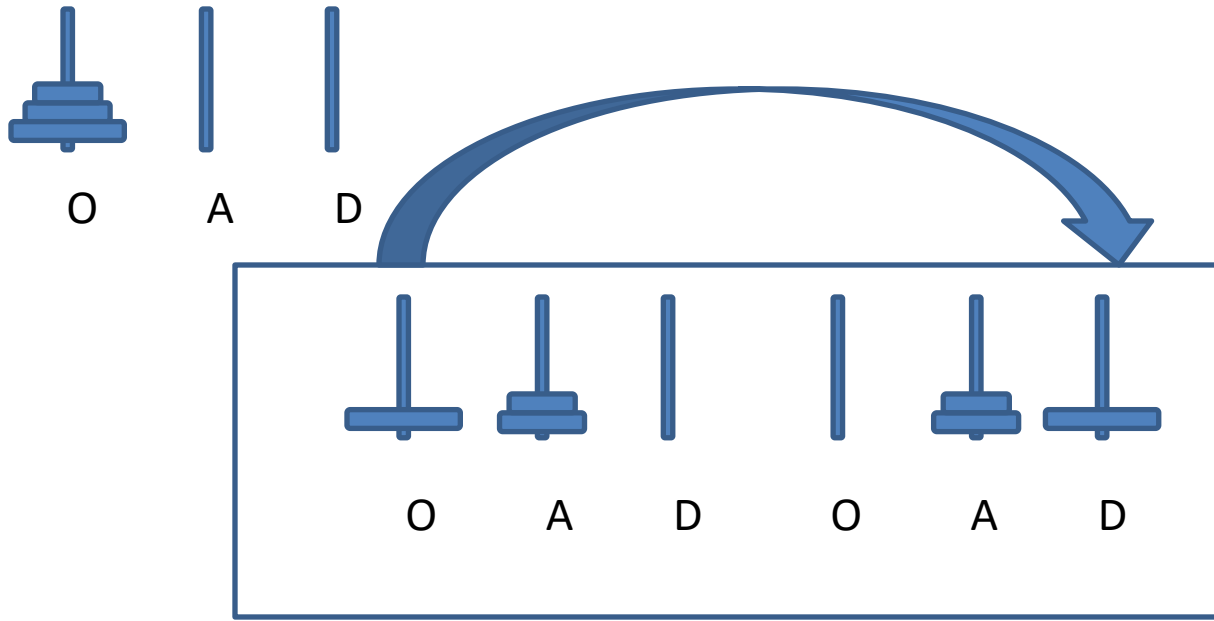
Todos os discos no poste auxiliar e o poste destino vazio.

=> Transferir todos os outros discos do poste original para o poste auxiliar
o poste Final funciona como auxiliar

Asseguir

=> Transferir todos os outros discos do poste auxiliar para o poste final
o poste Inicial funciona como auxiliar

Recursão



Algoritmo

```
Hanoi (n, postInicial, posteAuxiliar, posteFinal)
  Se n=1 Então
    MoveDisco(1, postInicial, posteFinal)
  Senão
    Hanoi (n -1, postInicial, posteFinal, posteAuxiliar)
    MoveDisco(n, postInicial, posteFinal)
    Hanoi (n -1 , posteAuxiliar, postInicial, posteFinal)
```

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n-1) + 1 & \text{se } n > 1 \end{cases}$$

VBA- Hanoi recursivo

```
Public Sub Hanoi(i As Integer, O As String, A As String, D As String)
    If i = 1 Then
        MsgBox ("mover disco " & i & " de " & O & " para " & A)
    Else
        Call Hanoi(i - 1, O, D, A)
        MsgBox ("mover disco " & i & " de " & O & " para " & A)
        Call Hanoi(i - 1, D, A, O)
    End If
End Sub
```

```
Public Sub jogo()
    Dim n As Integer
    Dim k As Integer
    Dim Origem As String
    Dim trabalho As String
    Dim destino As String
    Origem = "Origem"
    trabalho = "Trabalho"
    destino = "Destino"
    n = 3
    Call Hanoi(n, Origem, destino, trabalho)
End Sub
```

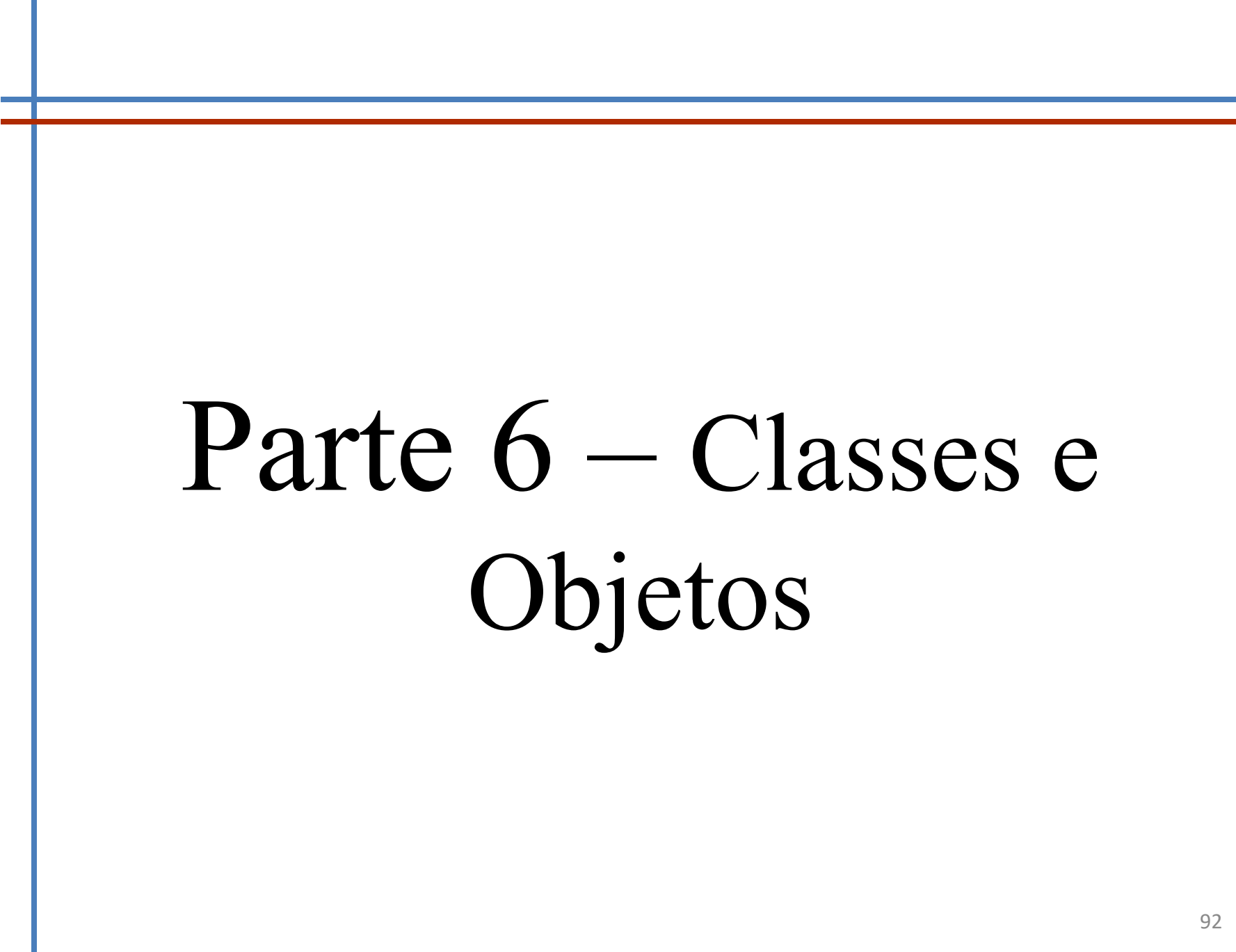
Número mínimo de movimentos

Resumindo para $n > 1$ tem-se

$$T(n) = 2^n - 1$$

ou seja, se os monges conseguirem um ritmo de um disco por segundo até que o mundo se desvaneça são necessários 2^{64} segundos, isto é cerca de 584942417 milénios

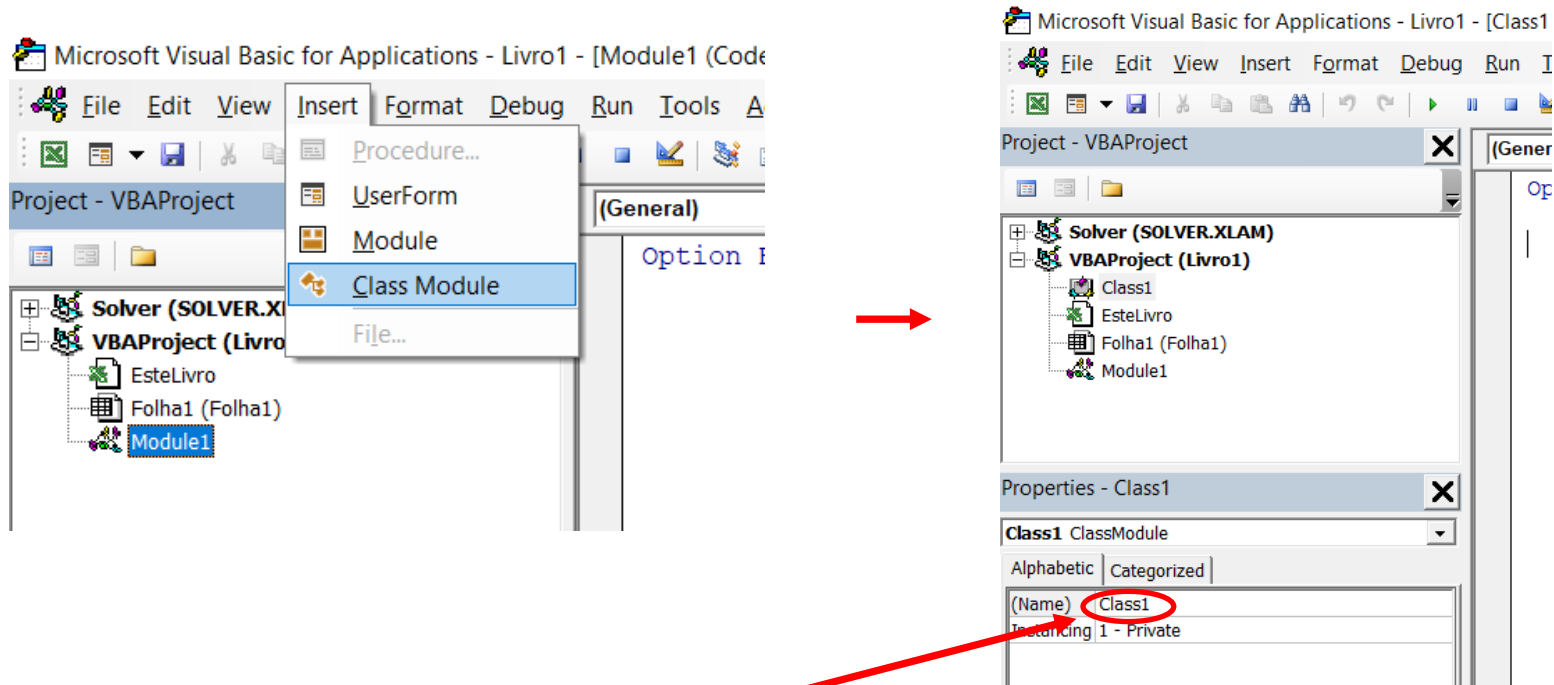
Podemos estar tranquilos!!!!



Parte 6 — Classes e Objetos

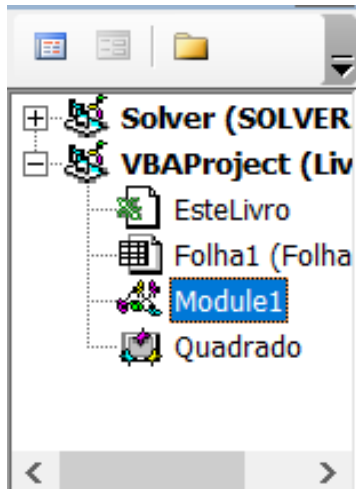
Classes

Uma **classe** é um tipo de dado estruturado com diversos **atributos (variáveis)** e **métodos (funções / procedimentos)** que alteram os seus atributos.



Definir o nome da classe

Classe Quadrado (exemplo)

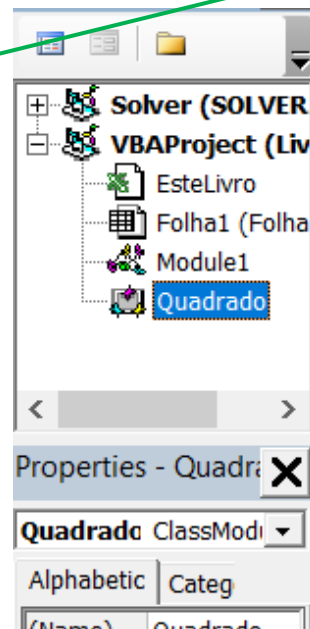


Option Explicit

```
Sub teste()  
    Dim Q1 As Quadrado  
    Set Q1 = New Quadrado  
  
    Q1.lado = InputBox("Lado do quadrado: ")  
    MsgBox ("A área é " & Q1.area)  
    MsgBox ("O perímetro é " & Q1.perimetro)  
End Sub
```

O objeto Q1 é
inicializado como
tendo lado igual 0

O ponto "." é usado para
acceder aos atributos e
métodos do objeto



Option Explicit

Public lado As Double

```
Private Sub Class_Initialize()  
    lado = 0  
End Sub
```

```
Public Function area()  
    area = lado ^ 2  
End Function
```

```
Public Function perimetro()  
    perimetro = 4 * lado  
End Function
```

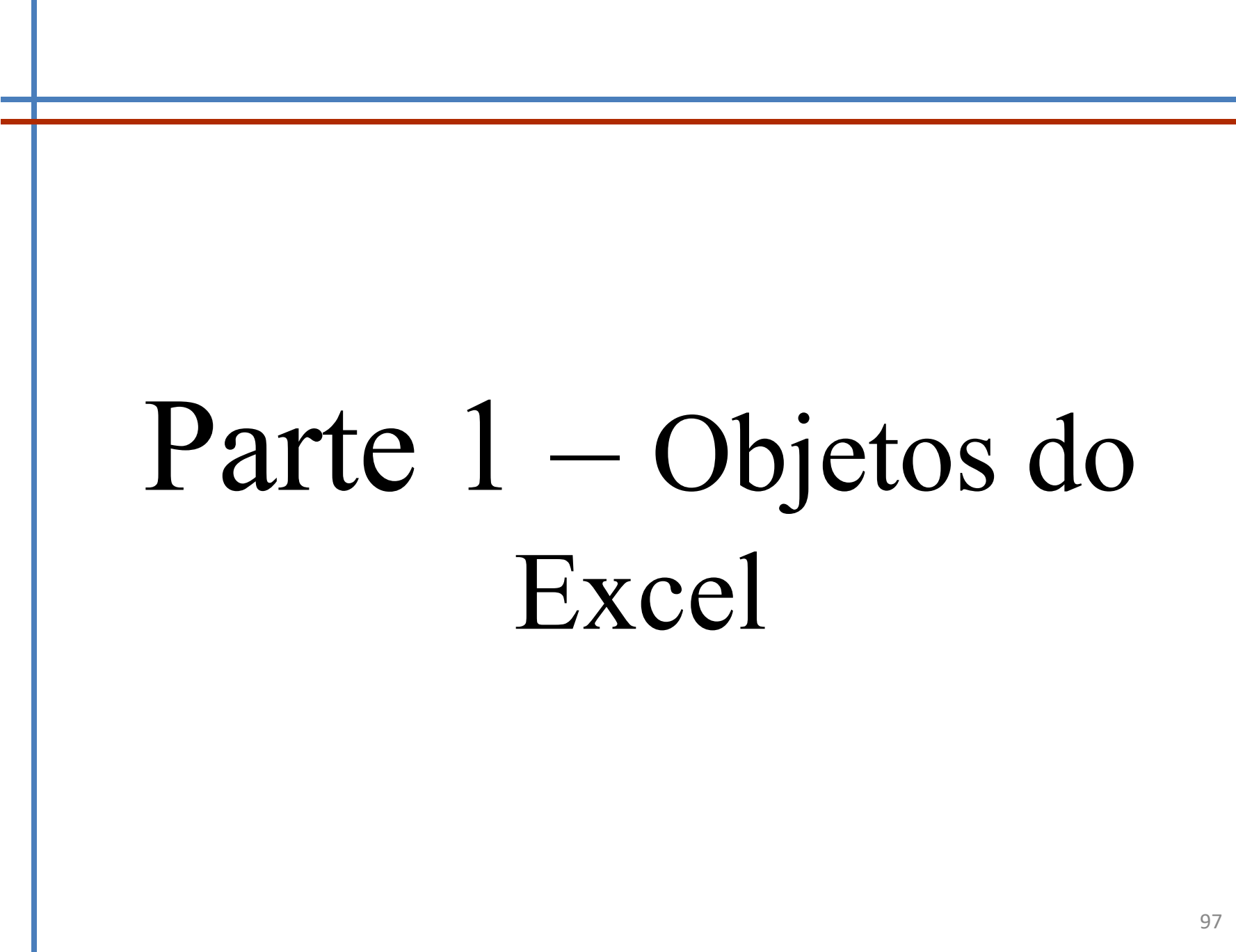
Classes e objetos

Um **objecto** é uma instância de uma **classe** e o conjunto dos atributos e métodos de uma classe são chamados **membros**.

Exemplo: A classe **Quadrado** tem 3 membros: um atributo ***lado*** e dois métodos ***area*** e ***perimetro***. Além disso, Q1 é um objeto do tipo **Quadrado**.

Capítulo II

Objetos do Excel e ambientes gráficos



Parte 1 — Objetos do Excel

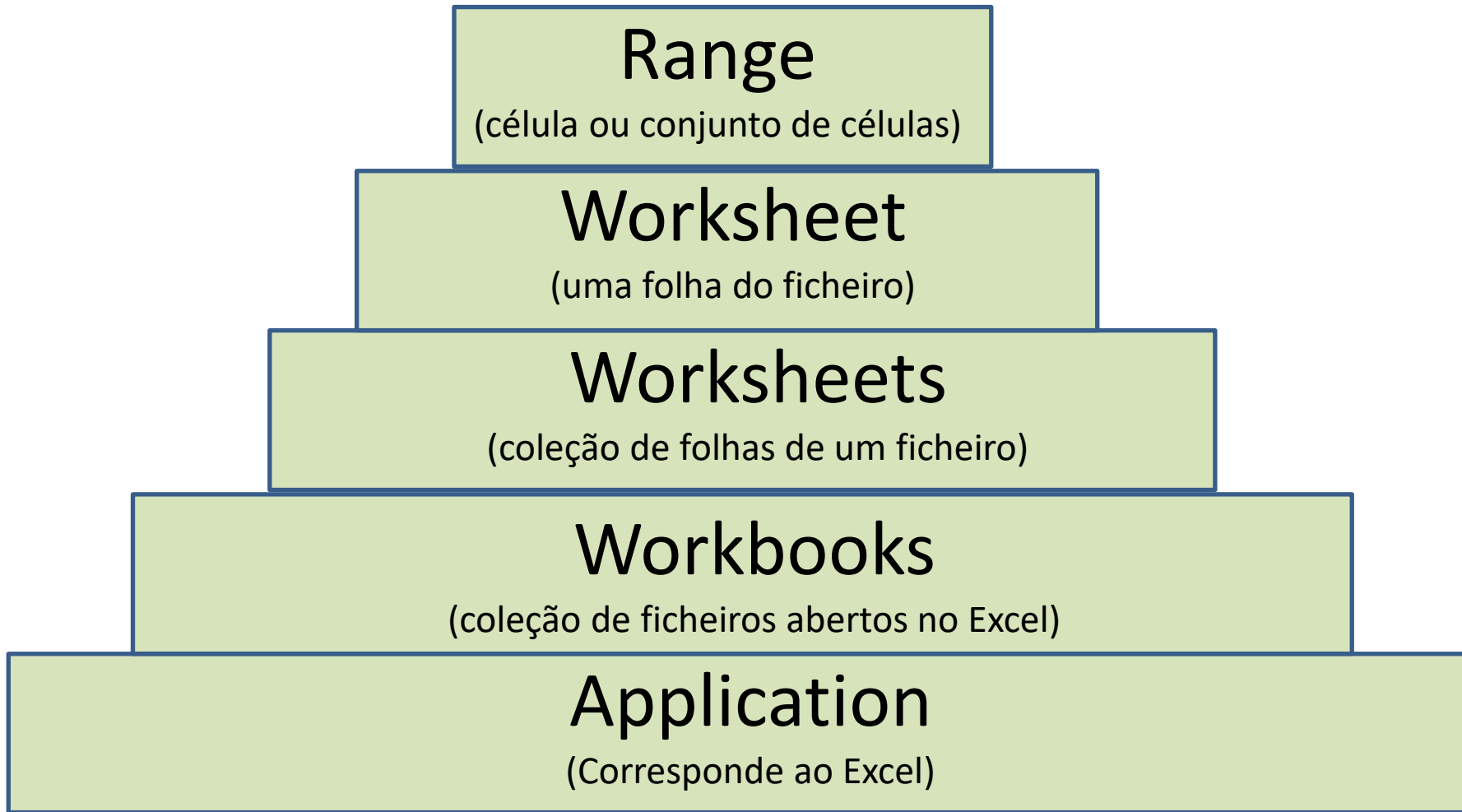
Objetos do Excel

Um **objeto** é tudo o que pode ser visto e manipulado de alguma forma, são componentes passíveis de serem programadas

Exemplos: Application, workbook, worksheet, Range,...

Também são objetos aqueles que permitem, construir uma interface gráfica: os **controles** que são colocados em janelas especiais designadas por **forms**.

Objetos (hierarquia)



Principais métodos do Excel

Nome do livro ativo no Excel

↳ `LivroAtivo = Application.ActiveWorkbook.Name`

Nome da folha do Excel ativa

↳ `FolhaAtiva = Workbooks(...).ActiveSheet.Name`

Tornar Ativa a folha do Excel com o nome “Dados”

↳ `Worksheets("Dados").Activate`

Mudar o nome da folha ativa para “NovoNome”

↳ `ActiveSheet.Name = “NovoNome”`

Principais métodos do Excel

Ler dados da folha ativa do Excel

- `valor = Application.Cells(1, 3)`
- `valor = ActiveSheet.Range("A3").Value`

Escrever dados na folha ativa do Excel

- `Application.Cells(1,3) = valor`
- `ActiveSheet.Range("A3").Value = valor`

Usar funções do Excel

- `media = Application.WorksheetFunction.Average(
Range(ActiveSheet.Cells(2, 2), ActiveSheet.Cells(12, 2)))`
- `media = Application.WorksheetFunction.Average(Range("E2", "E11"))`

Principais métodos do Excel

Criar nova folha ("FolhaNova") antes da que está ativa

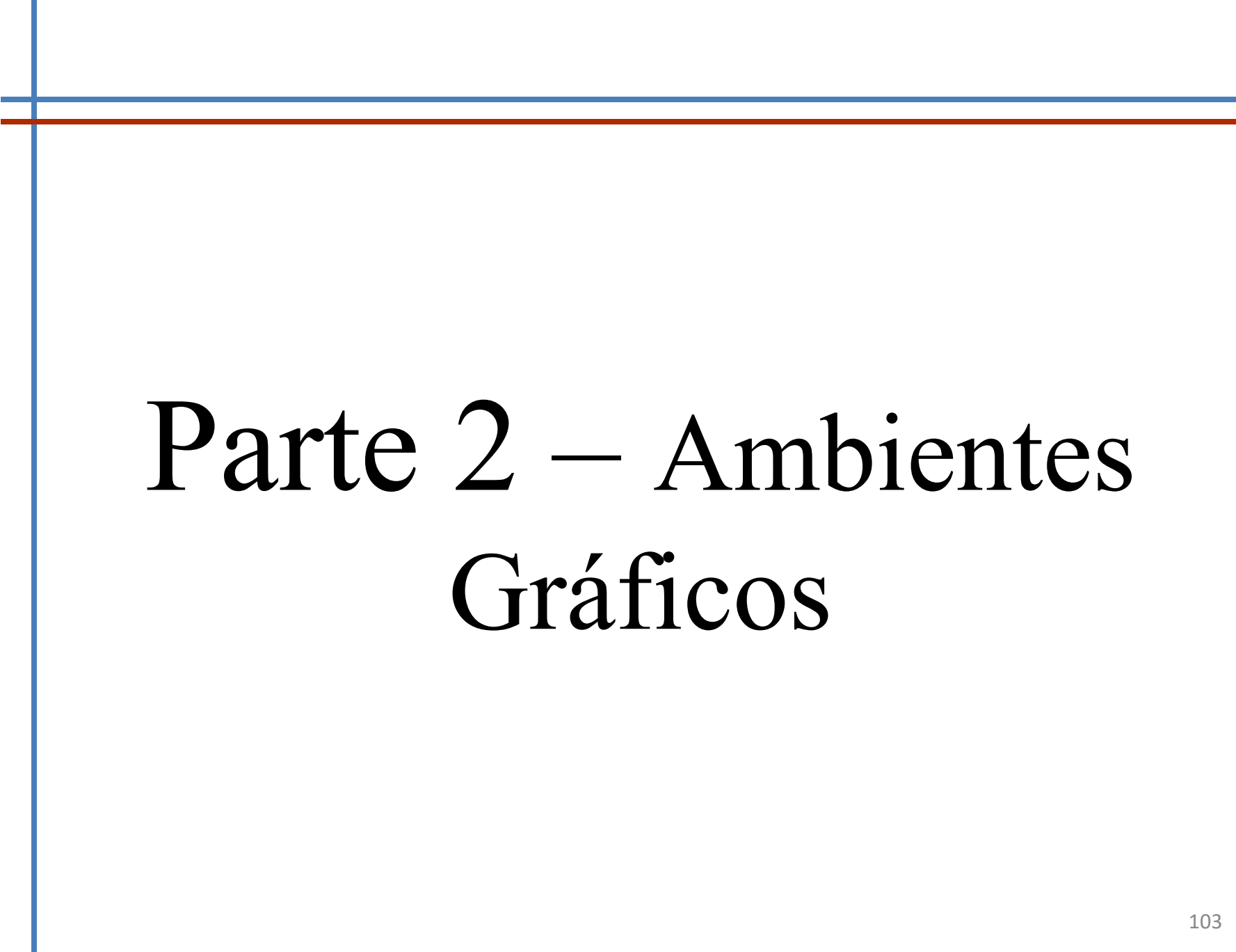
 *Worksheets.Add Before:=Worksheets(Worksheets.Count)*
ActiveSheet.Name = "FolhaNova"

Guarda o livro do Excel

 *Application.ActiveWorkbook.Save*

Sair do Excel

 *Application.Quit*



Parte 2 — Ambientes Gráficos

Comunicação com o utilizador

- i. Status bar
- ii. Message Boxes
- iii. Input Boxes
- iv. User forms

i. Status bar

- Mensagens simples podem ser visualizadas na barra de estado do programa enquanto algumas ações são executadas.

```
Application.DisplayStatusBar = True  
Application.StatusBar = "Please be patient..."  
    ' Instruções  
Application.StatusBar = False
```

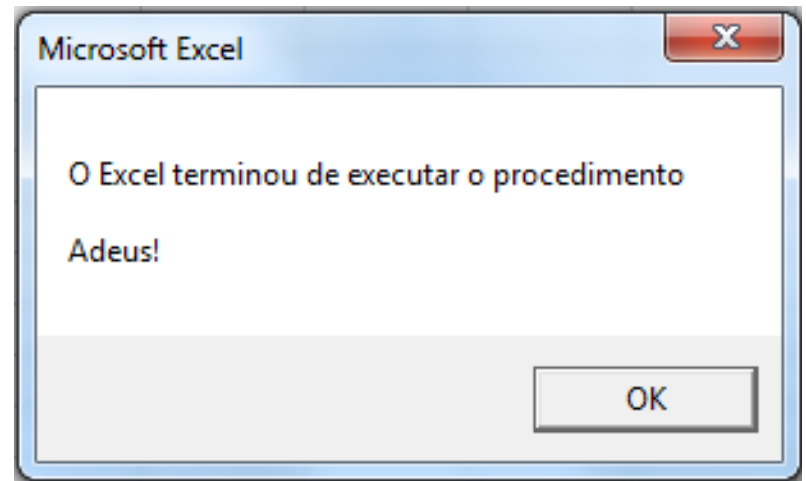
ii. Message Box

- Informar o utilizador sobre a execução de um procedimento
- Pedir ao utilizador que tome uma simples ação: Sim vs. Não
- Avisar o utilizador da existência de algum erro
- Etc...

ii. Message Box - simples

- Message Box simples

Dim mensagem **As** String



mensagem= "O Excel terminou de executar o procedimento" _
 & vbCr & vbCr & "Adeus!"

MsgBox mensagem

ii. Message Box - botões

Message Box com botões Sim/Não

Dim op **As** Long

op = MsgBox("Deseja formatar o computador?", **vbYesNo**)

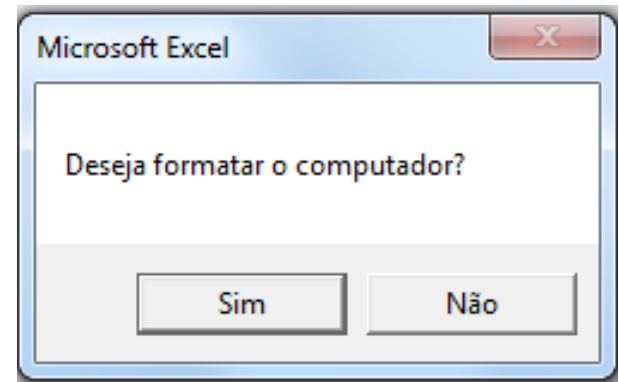
If op=**vbYes** **Then**

MsgBox "Contacte o seu programador"

Else

MsgBox "Computador não formatado"

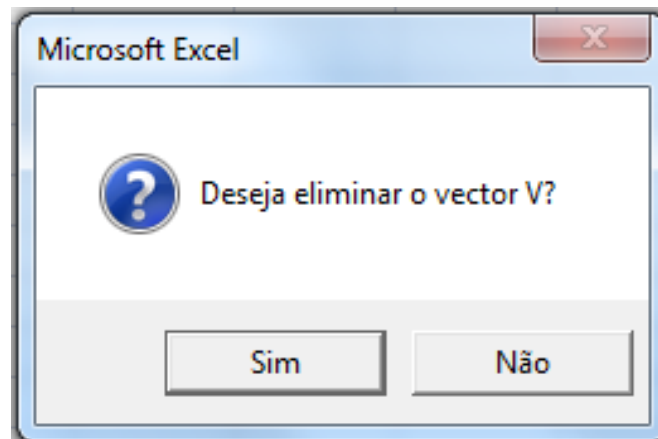
End if



ii. Message Box - Icones

- Colocar um icone numa MsgBox

op = MsgBox("Deseja eliminar o vector V?", **vbYesNo** + **vbQuestion**)



ii. Message Box - botões

TABLE 13.1: Message Box Types, Controlled by the *buttons* Argument

VALUE	CONSTANT	BUTTONS
0	<code>vbOKOnly</code>	OK
1	<code>vbOKCancel</code>	OK, Cancel
2	<code>vbAbortRetryIgnore</code>	Abort, Retry, Ignore
3	<code>vbYesNoCancel</code>	Yes, No, Cancel
4	<code>vbYesNo</code>	Yes, No
5	<code>vbRetryCancel</code>	Retry, Cancel

ii. Message Box - botões

TABLE 13.5: Constants for Selected Buttons

VALUE	CONSTANT	BUTTON SELECTED
1	vbOK	OK
2	vbCancel	Cancel
3	vbAbort	Abort
4	vbRetry	Retry
5	vbIgnore	Ignore
6	vbYes	Yes
7	vbNo	No

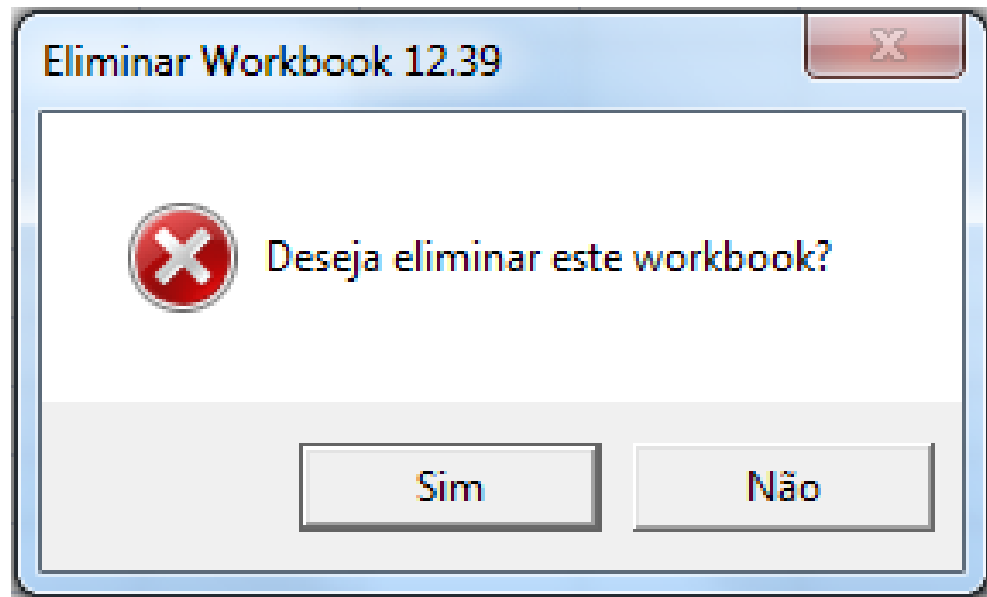
ii. Message Box - Icones

TABLE 13.2: Arguments for Message Box Icons

VALUE	CONSTANT	DISPLAYS
16	<code>vbCritical</code>	Stop icon
32	<code>vbQuestion</code>	Question mark icon
48	<code>vbExclamation</code>	Exclamation point icon
64	<code>vbInformation</code>	Information icon

ii. Message Box – Títulos

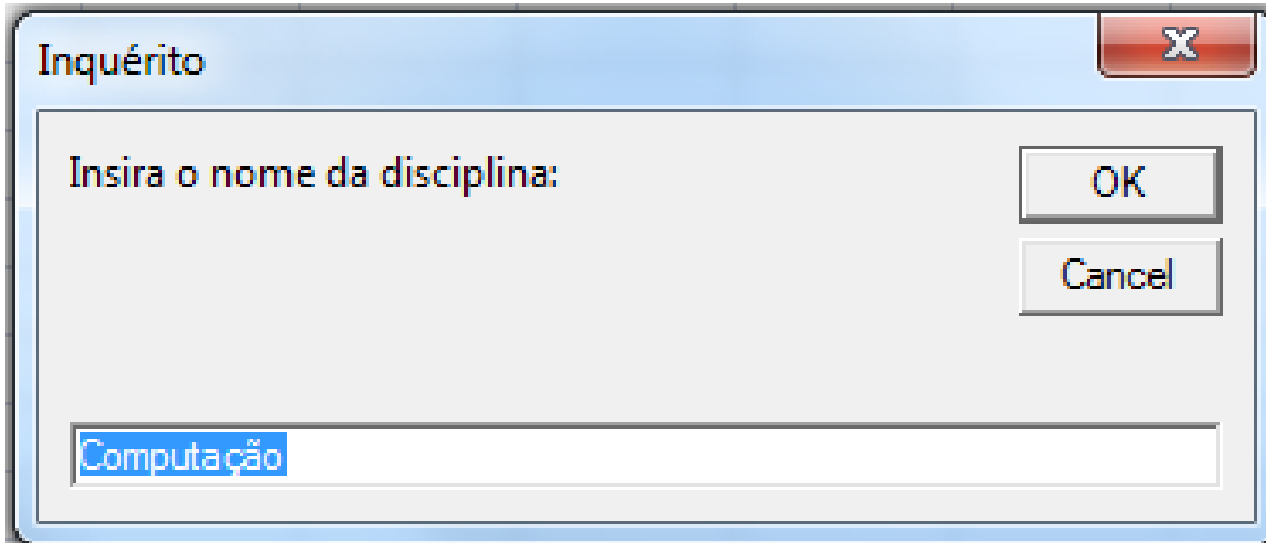
```
op = MsgBox("Deseja eliminar este workbook?", _  
           vbYesNo + vbCritical, "Eliminar Workbook 12.39")
```



iii. Input Box

Dim str As String

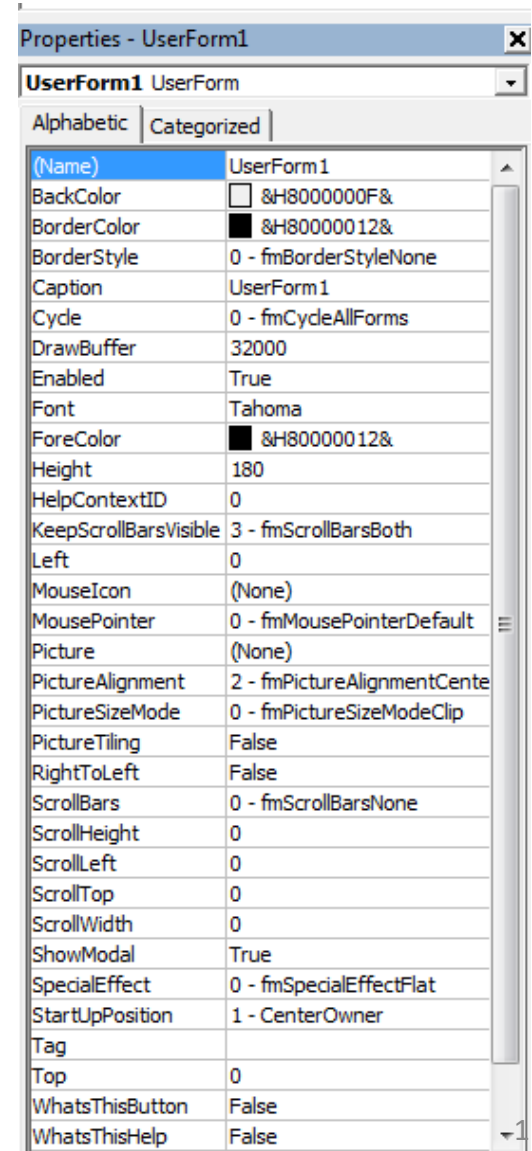
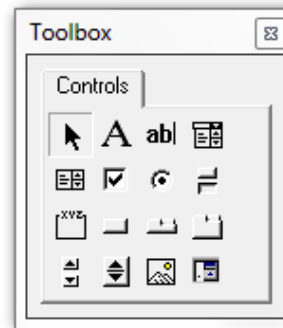
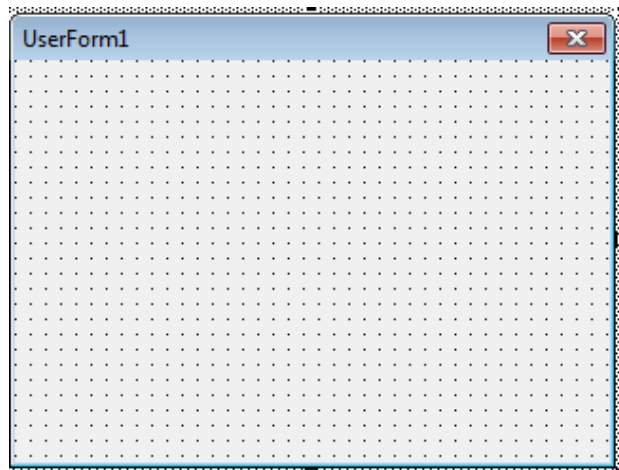
```
str = InputBox("Insira o nome da disciplina:", _  
               "Inquérito", "Computação")
```



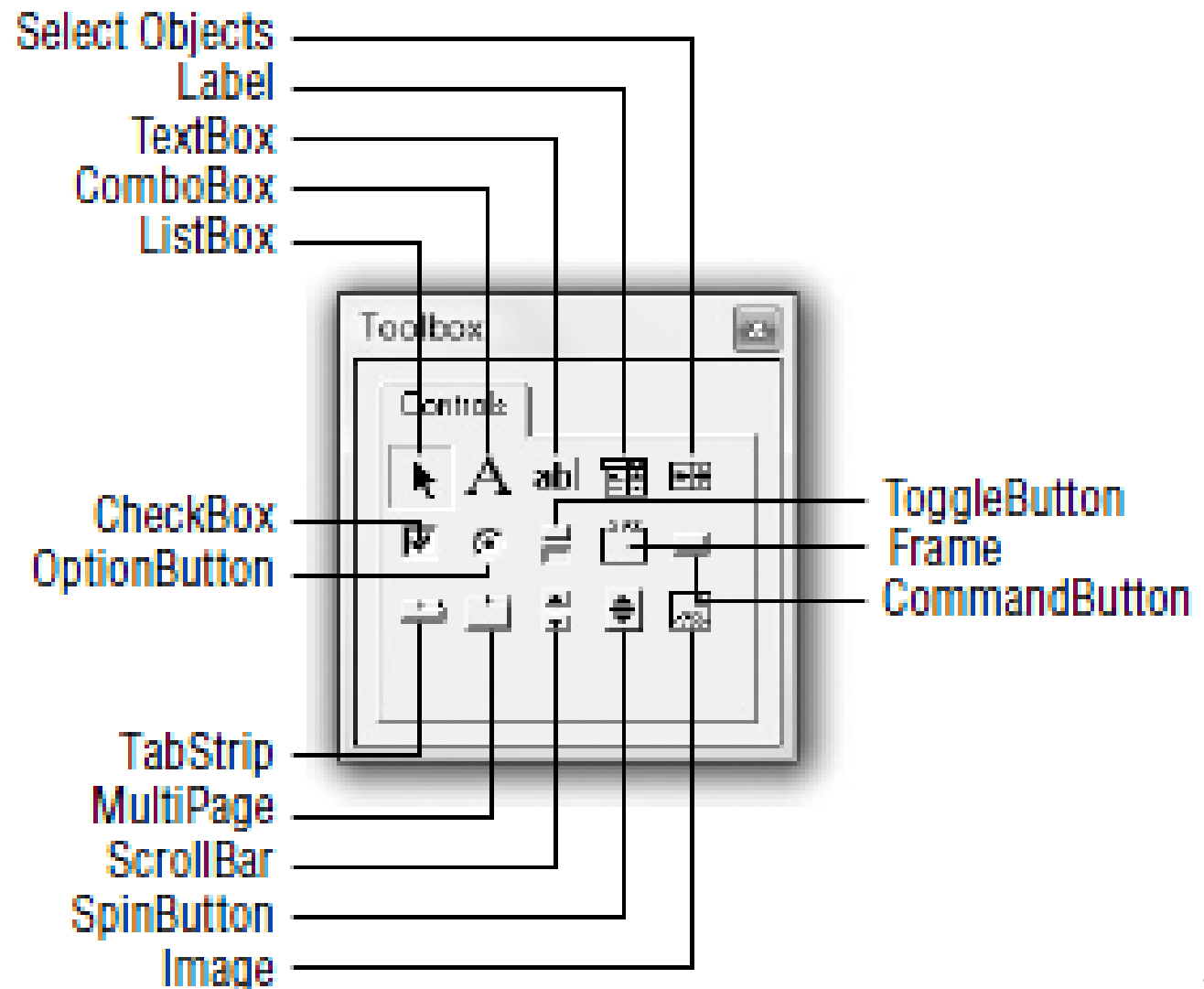
iV. UserForms

No developer:

- Insert // UserForm
- View // Toolbox
- View // Properties Window



Controles do UserForm



Descrição dos controlos

Select Objects

↳ Altera o ponteiro do rato para o modo de selecção.

Label

↳ Cria uma caixa de texto informativa.

TextBox

↳ Cria uma caixa de texto editável.

ListBox

↳ Cria uma lista de itens. O utilizador pode apenas escolher um item da lista.

ComboBox

↳ Cria um controlo que combina uma TextBox com uma ListBox

Descrição dos controlos

CheckBox

- ↳ Cria um quadrado com uma label que permite ao utilizador activar ou desactivar uma determinada opção.

OptionButton

- ↳ Cria um círculo com uma label que permite ao utilizador activar ou desactivar uma determinada opção.

ToggleButton

- ↳ Cria um Button com dois estados: On/Off

Frame

- ↳ Cria uma moldura com uma label que permite agrupar controlos.

Descrição dos controlos

CommandButton

- ↳ Cria um button que activa um certo comando/procedimento.

TabStrip

- ↳ Permite exibir diferentes conjuntos de informação para controles relacionados.

MultiPage

- ↳ Este controlo permite criar diversos userforms divididos por separadores. Cada separador tem o seu próprio layout.

ScrollBar

- ↳ Este controlo é pouco usado uma vez que ComboBox e ListBox vêm com ScrollBar.

Descrição dos controlos

SpinButton

- ↳ Este controlo consiste em duas setas (verticais ou horizontais) que permitem (por exemplo) incrementar ou decrementar valores numa TextBox.

Image

- ↳ Permite inserir uma imagem no userform.

UserForms

- *Load /Unload* (carrega/descarrega userform da memória do computador)

Load *Form1*

Unload *Form1*

- *Show/Hide* (mostra/esconde userform)

Sub programa()

Load form1 'Carrega a form1 para a memória do computador

form1.Show 'mostra a form1

End Sub