



# Informática Industrial

Aula Prática 1

## Programação em Visual Basic

José Paulo Santos, Miguel Riem de Oliveira, Jorge Almeida

**É obrigatório o envio do código do trabalho prático, presente no final de cada guião de aula prática, até à data da aula prática da semana seguinte. Ou seja, o trabalho prático tem de estar obrigatoriamente submetido no elearning antes da data da aula prática + 1 semana. Para este efeito, conta a aula prática em que o aluno estiver inscrito. Para submeter deve comprimir a pasta do projeto num ficheiro “rar” e fazer o upload.**

**O não cumprimento implicará a anulação da nota do respetivo questionário.**

**Não são aceites trabalhos enviados por email, apenas submissões no elearning.**

## 2. INTRODUÇÃO AO VISUAL BASIC

Longe vão os dias em que era necessário usar um editor de texto genérico para escrever o código e para o gravar num ficheiro em disco. Depois era necessário chamar um programa (compilador) que abria esse ficheiro de texto e gerava um segundo ficheiro com o código máquina (obj), depois o programador necessitava de chamar um terceiro programa “Linker” para juntar os vários ficheiros de código máquina gerados pelo compilador, com outros ficheiros com código máquina (lib) fornecidos pela empresa do “compilador”, para que finalmente fosse possível gerar em disco um ficheiro executável que fazia o que o programador pretendia. (quando tudo corria bem...)

Hoje em dia existem programas comerciais, como o *Visual Basic 2017* que integram todas estas funcionalidades, permitindo que o programador a partir de um ambiente integrado de desenvolvimento (IDE), possa editar vários ficheiros com o código fonte, associá-los a um mesmo projeto, compilar todos estes ficheiros, linkar e executar o programa sempre no mesmo ambiente. (*secção 2.2 Ambiente Integrado de desenvolvimento*).

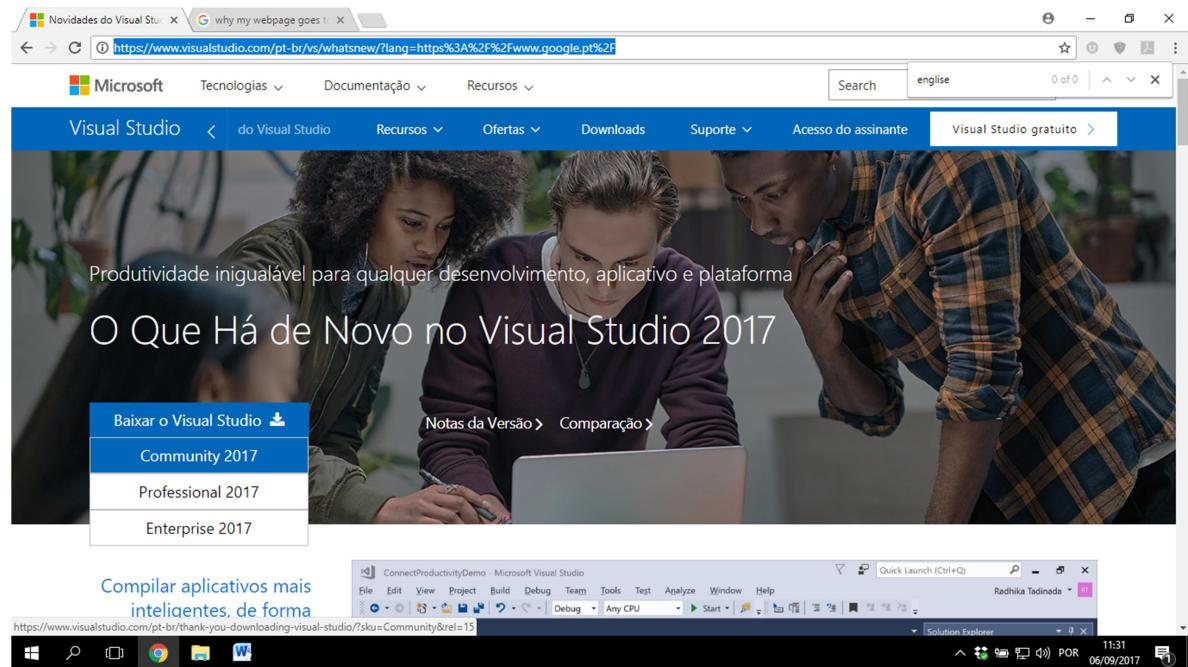
Por outro lado, todos os programas desenvolvidos por um programador necessitam de ter uma interface para que o utilizador possa interagir com o programa e visualizar os resultados. Uma das maiores vantagens do Visual Basic é a enorme facilidade e rapidez com que um programador pode criar novos programas com interfaces gráficas amigáveis (*secção 2.3 Construção da interface gráfica*). Naturalmente que a interface gráfica só por si não chega, é necessário saber desenvolver o código necessário para processar os dados recebidos e gerar os resultados (*secção 2.4 Linguagem de programação*)

## 2.1 Instalar o Visual Basic 2017

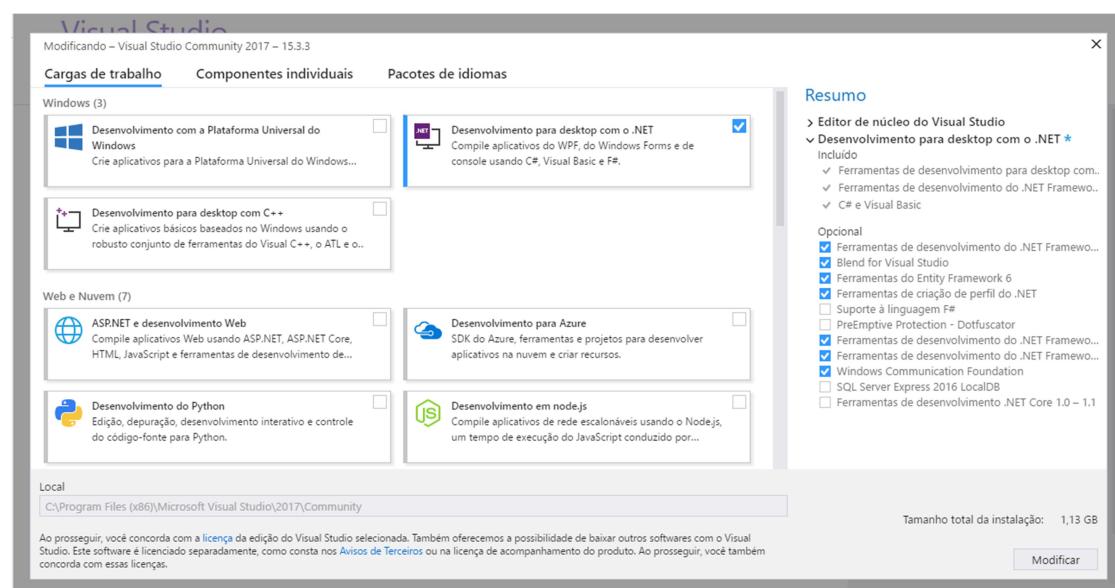
O Visual Basic 2017 é gratuito e pode ser transferido a partir do sítio da Microsoft

<https://www.visualstudio.com/pt-br/vs/whatsnew/?lang=https%3A%2F%2Fwww.google.pt%2F>

Deve-se escolher a opção **baixar** e depois **Visual Studio Community 2017**.



Depois de iniciar a instalação, escolher a instalação do *Workload .NET desktop development*.



De facto, o Visual Basic 2017 faz parte de um pacote mais completo conhecido por **Visual Studio Community 2017** que contém também suporte para outras linguagens, por exemplo: Visual C#, Visual C++ ou Visual Web Developer.

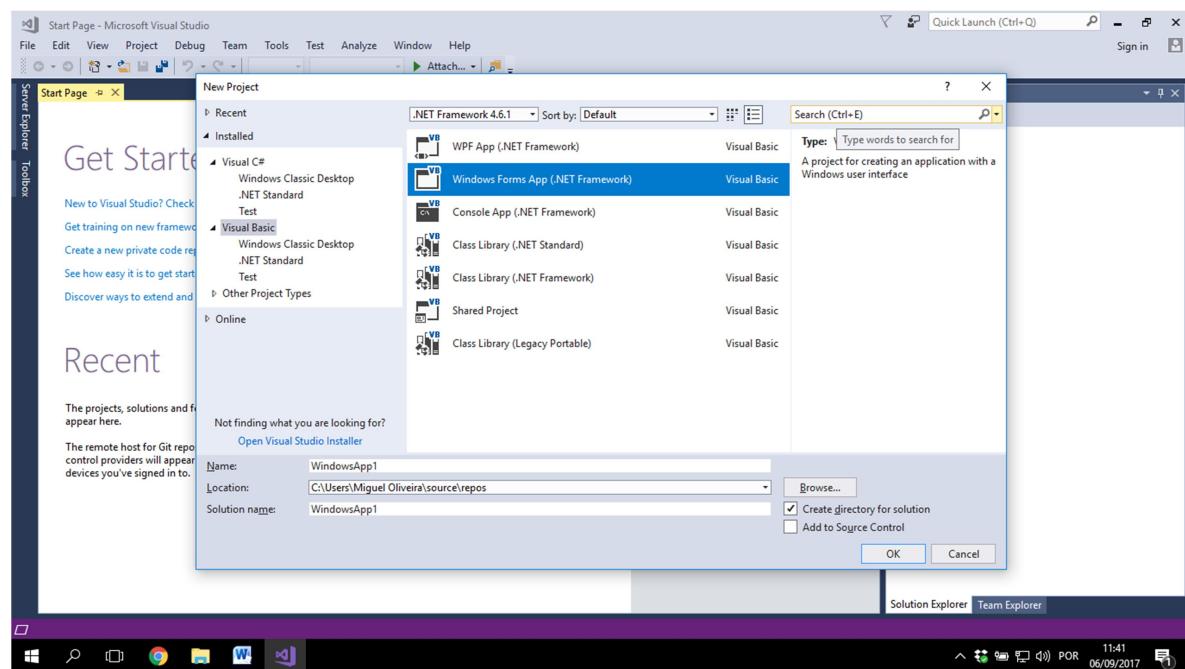
No âmbito de Informática Industrial iremos utilizar apenas o *Visual Basic*, que passa a ser referido por *VBasic*. Para encontrar informação adicional (documentação, exemplos de código e outros) experimente os dois links:

<http://msdn.microsoft.com/en-us/vbasic/default.aspx>

[http://msdn.microsoft.com/query/dev10.query?appId=Dev10IDEF1&l=EN-US&k=k\(MSDNSTART\)&rd=true](http://msdn.microsoft.com/query/dev10.query?appId=Dev10IDEF1&l=EN-US&k=k(MSDNSTART)&rd=true)

## 2.2 Ambiente integrado de desenvolvimento

Para criar um novo projeto que permita desenvolver uma aplicação Windows, com interface, deve selecionar a opção “File – New Project” e selecionar a linguagem Visual Basic e a subopção “Windows Forms App” como a figura seguinte ilustra.

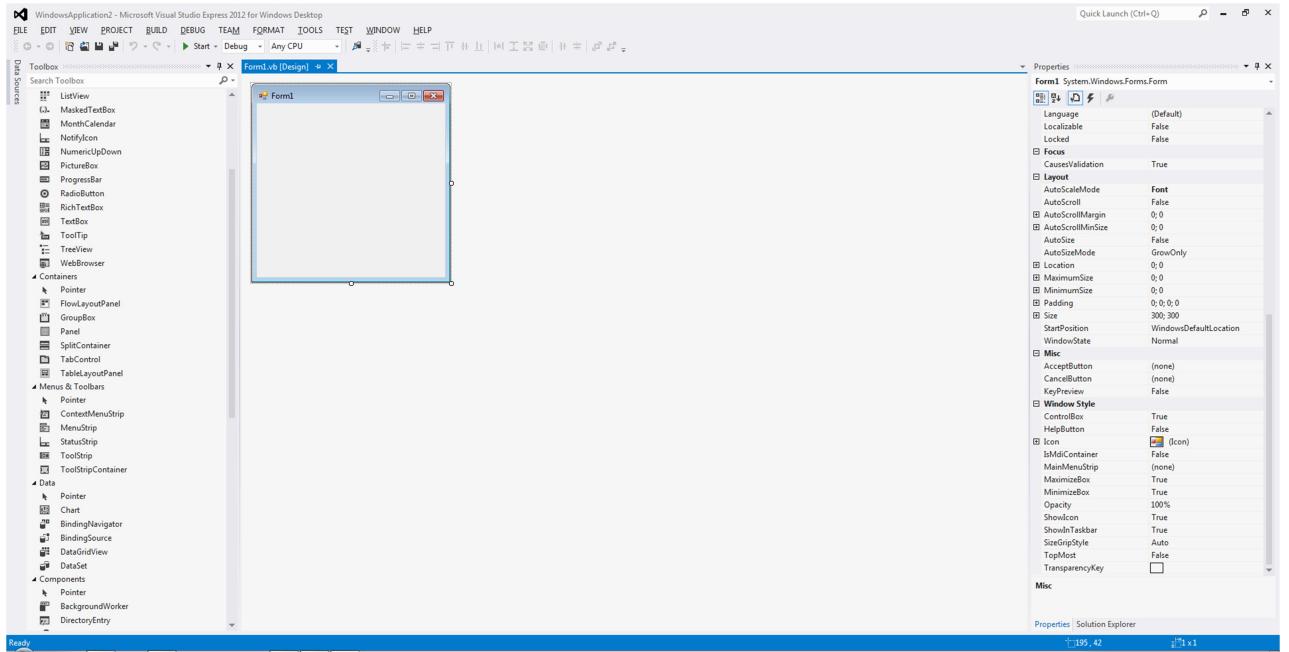


A figura seguinte apresenta a janela inicial do ambiente integrado de desenvolvimento IDE.

Ao centro aparece a janela **Form1**, à esquerda aparece a janela **Toolbox** onde é possível ir buscar os objetos gráficos para colocar na janela Form1 (botões, caixas de texto,...), à direita aparece a janela das **Properties**.

Todos os objetos gráficos (controlos) que colocarmos no Form, e o próprio Form, têm propriedades (cor, localização no ecrã, nome,...). Sempre que seleccionarmos um objecto do **Form** “clicando” sobre ele, a janela **Properties** apresenta as propriedades do objecto (controlo) selecionado.

Se alguma destas janelas não aparecer no ecrã, pode selecionar o menu **View**, a opção **OtherWindows** e depois selecionar a(s) janelas que pretender ver : **Toolbox**, **Solution Explorer**, **Properties Window**, ...



Para mais informações sobre o IDE deve consultar o livro recomendado.

## 2.3 Construção da interface gráfica dos programas

Uma das maiores vantagens do *VBasic* reside precisamente na facilidade e rapidez com que é possível desenvolver uma interface gráfica para o nosso programa.

Na janela “**Toolbox**” temos à disposição muitos “**Controlos**” que podemos “arrastar” para a janela “**Form**”. É esta janela que o utilizador irá ver quando executar o nosso programa.

Nesta secção iremos estudar vários controlos, este estudo baseia-se no princípio que “uma imagem vale mil palavras” por isso iremos analisar vários exemplos práticos. Será apresentado o código correspondente a cada exemplo, e estudado cada um dos controlos usados nesses exemplos.

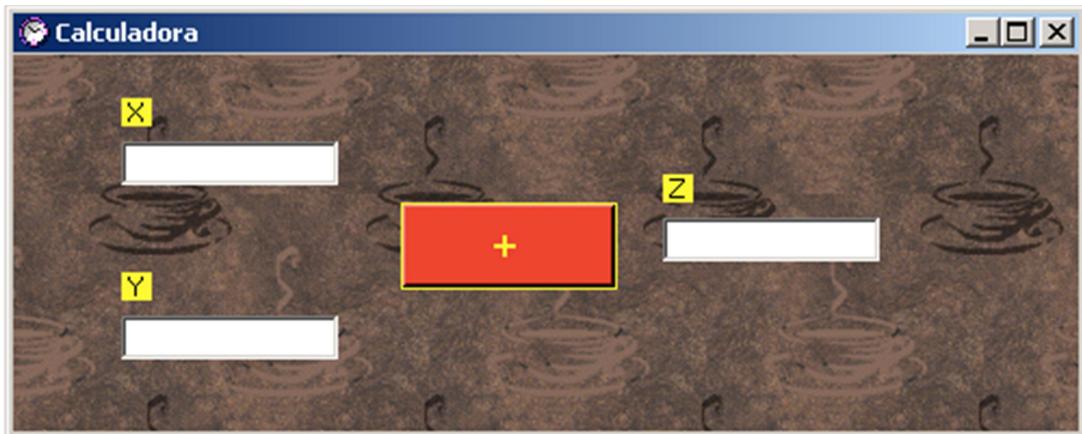
No final desta secção, depois de realizar todos os exemplos, o leitor terá implementado um editor de texto com várias janelas/Forms (interfaces) e terá usados todos os controlos que a seguir se enumeram:

*Form, Label, TextBox, Button, RadioButton, PictureBox, Combobox, OpenFileDialog, SaveFileDialog, StreamWriter, StreamReader.*

### 2.3.1 Exemplo “Calculadora\_soma”

Vamos começar por um exemplo simples, intitulado Calculadora\_Soma. Este programa permite calcular a soma de dois valores. O utilizador introduz dois valores, um em X, o outro em Y e quando premir o botão vermelho (+) aparece o resultado da soma em Z (figura seguinte).

Nesta interface **Form1** o programador utiliza três controlos do tipo **Label**, três controlos do tipo **Textbox** e um controlo do tipo **Button**.



'Nesta classe é definido todo o código necessário para a execução da interface (Janela) Form1  
Public Class Form1

' A subrotina Form1\_Load é executada automaticamente, uma vez, quando a janela Form1 aparece no écran

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

```
    Me.Text = "Calculadora"           ' A palavra Calculadora aparece na parte de cima da janela
    Form1                           ' A janela Form1 ocupa 200 por 500 pixeis no écran
    Me.Height = 200
    Me.Width = 500
    Me.BackgroundImage = Image.FromFile("c:\windows\coffeebean.bmp") ' Esta passa a ser a
    imagem de fundo da janela Form1
    Me.Icon = New Icon("Alarm.ico")   ' Esta imagem "icon" aparece no canto superior da janela
    Form1                           ' A Textbox1 ocupa 100 por 30 pixeis dentro da Janela
    Form1                           ' Esta Textbox1 tem as coordenadas X=50,Y=40
    TextBox1.Width = 100
    TextBox1.Height = 30
```

TextBox1.Location = New Point(50, 40) ' A caixa de texto TextBox1, o seu canto superior
 esquerdo, tem as coordenadas X=50,Y=40

TextBox1.Width = 100 ' A Textbox1 ocupa 100 por 30 pixeis dentro da Janela
 Form1 ' A Textbox1 tem as coordenadas X=50,Y=40
 TextBox1.Height = 30

TextBox2.Location = New Point(50, 120)

TextBox2.Width = 100

TextBox2.Height = 30

TextBox3.Location = New Point(300, 75)

TextBox3.Width = 100

TextBox3.Height = 30

Button1.Text = "+"

Button1.BackColor = Color.Red ' O botao passa a ter a cor de fundo vermelha
 Button1.ForeColor = Color.Yellow ' e o texto a amarelo

Button1.Width = 100

Button1.Height = 40

Button1.Visible = True

Label1.Text = "X"

Label1.BackColor = Color.Yellow ' A cor de fundo do Label1 passa a amarelo

```

Label1.ForeColor = Color.Black      ' e o texto que aparece no Label1, neste caso a letra X,
passa preto
Label1.Width = 100
Label1.Height = 40
Label1.Location = New Point(50, 20)
Label1.Visible = True

Label2.Text = "Y"
Label2.BackColor = Color.Yellow
Label2.ForeColor = Color.Black
Label2.Width = 100
Label2.Height = 40
Label2.Location = New Point(50, 100)
Label2.Visible = True

Label3.Text = "Z"
Label3.BackColor = Color.Yellow
Label3.ForeColor = Color.Black
Label3.Width = 100
Label3.Height = 40
Label3.Location = New Point(300, 55)
Label3.Visible = True

End Sub

```

*' Quando o utilizador do programa permitir o botao (+) da interface Form1 esta subrotina será executada*

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Os conteúdos das caixas de texto da esquerda são convertidos em números inteiros
    ' e o resultado da sua soma é visualizado na caixa de texto da direita (textBox3)
    TextBox3.Text = CStr(CInt(textBox1.Text) + CInt(textBox2.Text))
End Sub

```

End Class

Depois de apresentarmos a interface e o código, vamos analisar com maior detalhe as propriedades de cada um dos controlos/objetos utilizados na interface.

### Objeto “Form1”

Altere o titulo da janela “Form1” para “Calculadora”	Form1.Text = "Calculadora"
Altere a altura da janela “Form1” para 200 pixels	Form1.Height = 200
Altere a largura da janela “Form1” para 500 pixels	Form1.Width = 500
Altere a imagem do canto superior esquerdo da janela Form1	Form1.Icon = New Icon("Calc.ico")
Altere a imagem de fundo da janela Form1	 
	Form1.BackgroundImage =
Image.FromFile("C:\windows\café.bmp")	

### Objeto “Label1”

Arraste um controlo do tipo *“Label”* para a janela “Form1”. A partir desse momento passa a ter um objeto *“Label1”* visível na interface. Este controlo tem o aspecto de

uma linha de texto, permite ao utilizador ler as mensagens de texto que o programador previamente definiu.

Escreva a mensagem “Caixa de texto” na caixa de texto	Label1.Text = "X"
Altere a cor de fundo da janela para amarelo	Label1.ForeColor = Color.Yellow
Altere a cor de fundo da janela para amarelo	Label1.BackColor = Color.Grey
Altere a altura da caixa de texto para 20 pixels	Label1.height = 50
Altere a largura da caixa de texto para 20 pixels	Label1.Width = 100
Altere o nome deste objeto para “txtJanela”	Label1.Name = "lbX"

### *Objeto “TextBox1”*

Arraste um controlo do tipo “TextBox” para a janela “Form1”. A partir desse momento passa a ter uma caixa de texto com o nome “*TextBox1*”. Este controlo tem o aspecto de uma caixa de texto, permite ao utilizador e ao programador escrever ou ler mensagens de texto nela.

Escreva a mensagem “Caixa de texto” na caixa de texto	TextBox1.Text = "Caixa de texto"
Altere a cor de fundo da janela para amarelo	TextBox1.BackColor = Color.Yellow
Adicione barras de deslocamento Horiz,Vert “ScrollBar”	TextBox1.ScrollBars = ScrollBars.Both
Janela com várias linhas	TextBox1.Multiline = True
Altere a localização da textbox	TextBox1.Location = New Point(50,40)
Altere a altura da caixa de texto para 20 pixels	TextBox1.height = 50
Altere a largura da caixa de texto para 20 pixels	TextBox1.Width = 100
Alinhe ao centro o texto que será escrito na caixa	TextBox1.TextAlign= HorizontalAlignment.Center
Altere o nome deste objeto para “txtJanela”	TextBox1.Name = "txtJanela"

Note que depois de alterar o nome do objeto, se quiser alterar alguma das suas propriedades deve fazer por exemplo *txtJanela.Width = 400*

### *Objeto “Button1”*

Arraste um controlo do tipo “Button” para a janela “Form1”. A partir desse momento passa a ter um botão com o nome “*Button1*”. Este controlo permite ao utilizador “clicar” sobre o botão fazendo com que o código associado a este botão seja executado.

Escreva a mensagem “Soma” na propriedade	Button1.Text = "Soma "
Altere a cor de fundo do botão para amarelo	Button1.BackColor = Color.Yellow
Altere a cor da letra para azul	Button1.ForeColor = Color.Blue
Altere a altura da caixa de texto para 20 pixels	Button1.height = 50
Altere a largura da caixa de texto para 20 pixels	Button1.Width = 100
Altere o nome deste objeto para “BtSoma”	Button1.Name = "BtSoma"

### **2.3.2 Exemplo “Seleção de imagens”**

Neste exemplo, o utilizador pode selecionar uma de várias imagens na interface **Form1** do programa. O utilizador selecciona um de três **Radiobutton**, fazendo aparecer a imagem respectiva na **Picturebox** existente na janela.



Public Class Form1

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    RadioButton1.Location = New Point(50, 50)
    RadioButton1.Width = 50
    RadioButton1.Height = 25
    RadioButton1.Text = "Bolhas"           ' Bolhas é o nome que o utilizador vê
```

```
    With RadioButton2
        .Location = New Point(50, 70)
        .Width = 50
        .Height = 25
        .Text = "Pedra"
        .Select()                  ' Este radiobutton fica selecionado
        .BackColor = Color.Aquamarine ' Esta é a cor de fundo do Radiobutton2
    End With
```

```
    With RadioButton3
        .Location = New Point(50, 90)
        .Width = 50
        .Height = 25
        .Text = "Salmao"
    End With
```

```
    PictureBox1.Location = New Point(200, 20)
    PictureBox1.Anchor = AnchorStyles.Left
    PictureBox1.BackColor = Color.AliceBlue
    PictureBox1.BorderStyle = BorderStyle.Fixed3D
```

```
End Sub
```

**' Este subprocedimento é executado quando o utilizador selecionar o Radiobutton1**

```
Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ...
```

```
    RadioButton1.BackColor = Color.Aquamarine           ' A cor de fundo do radiobutton1 passa a
    ser Aquamarine
```

```
    PictureBox1.BackgroundImage = Image.FromFile("c:\windows\Soap Bubbles.bmp") ' Na
    picturebox1 aparece a imagem Soap Bubbles
```

```
End Sub
```

```

Private Sub RadioButton2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ...
    RadioButton2.BackColor = Color.Aquamarine
    PictureBox1.BackgroundImage = Image.FromFile("c:\windows\Greenstone.bmp")
End Sub

Private Sub RadioButton3_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ...
    RadioButton3.BackColor = Color.Aquamarine
    PictureBox1.BackgroundImage = Image.FromFile("c:\windows\santa fe stucco.bmp")
End Sub

End Class

```

### *Objeto RadioButton1*

---

Arraste um controlo do tipo “*RadioButton*” para a janela “Form1”. A partir desse momento passa a ter um botão com o nome “*RadioButton1*”. Este controlo permite ao utilizador “clicar” sobre o botão fazendo com que o código associado a este botão seja executado.

Escreva a mensagem “ <i>Soma</i> ” na propriedade	RadioButton1. <b>text</b> = “ <i>Bolhas</i> ”
Altere a cor de fundo do botão para amarelo	RadioButton1. <b>BackColor</b> = <i>Color.Yellow</i>
Altere a cor da letra para azul	RadioButton1. <b>ForeColor</b> = <i>Color.Blue</i>
Altere a altura da caixa de texto para 20 pixels	RadioButton1. <b>height</b> = 50
Altere a largura da caixa de texto para 20 pixels	RadioButton1. <b>Width</b> = 100
Altere a largura da caixa de texto para 20 pixels	RadioButton1. <b>Position</b> = new Point(200,20)
Altere a largura da caixa de texto para 20 pixels	RadioButton1. <b>Width</b> = 100
Altere o nome deste objeto para “BtSoma”	RadioButton1. <b>Name</b> = “ <i>RbBolhas</i> ”

### *Objeto PictureBox1*

---

Arraste um controlo do tipo “*PictureBox*” para a janela “Form1”. A partir desse momento passa a ter um botão com o nome “*PictureBox1*”. Este controlo permite ao utilizador “clicar” sobre o botão fazendo com que o código associado a este botão seja executado.

Altere a cor de fundo do botão para amarelo	<i>PictureBox1.BackColor</i> = <i>Color.Yellow</i>
Altere a altura da caixa de texto para 20 pixels	<i>PictureBox1.height</i> = 50
Altere a largura da caixa de texto para 20 pixels	<i>PictureBox1.Width</i> = 100
Altere a posição do canto superior esquerdo	<i>PictureBox1.Location</i> = new Point(200,20)
A picturebox acompanha o lado esquerdo do form	<i>PictureBox1.Anchor</i> = AnchorStyles.Left
Altere a moldura exterior da picture box	<i>PictureBox1.BorderStyle</i> = BorderStyle.Fixed3D
Altere a imagem presente na picturebox	<i>PictureBox1.BackGroundImage</i> =Image.FromFile(“c:\windows\Gre.bmp”)

Altere a imagem presente na picturebox PictureBox1.BackGroundImageLayout=ImageLayout.Stretch  
Altere o nome deste objeto para PictureBox1.Name = "pBBolhas"  
"pBBolhas"

### 2.3.3 Exemplo “Lista de String”

Neste exemplo pretende-se apresentar o controlo **ComboBox**.

O texto que o utilizador introduzir na **TextBox** da esquerda será adicionado ou removido da **ComboBox** da direita quando o utilizador premir o botão adicionar ou remover, respectivamente.

Nesta interface **Form1** o programador utiliza cinco controlos do tipo **Label**, três controlos do tipo **Textbox**, três controlo do tipo **Button** e um controlo do tipo **ComboBox**.



Public Class Form1

' Quando o utilizador clicar no botao Adicionar esta subrotina é chamada

```
Private Sub BtAdicionar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtAdicionar.Click
    ComboBox1.Items.Add(txtItem.Text)
    txtNItens.Text = ComboBox1.Items.Count
    ' adiciona à combobox o texto presente na textbox
    ' visualiza na textbox o nº de linhas/itens
    ' presentes na combobox
End Sub
```

' Quando o utilizador clicar no botao Remover esta subrotina é chamada

```
Private Sub btRemover_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btRemover.Click
    ComboBox1.Items.Remove(txtItem.Text)
    ' remove da combobox o item/linha/texto
    ' presente na textbox
    txtNItens.Text = ComboBox1.Items.Count
End Sub
```

' Quando o utilizador selecionar na combobox uma nova linha esta subrotina é chamada

```
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles ...
    txtItemSelecionado.Text = ComboBox1.SelectedItem ' Visualiza a linha/item selecionado pelo
    ' utilizador
End Sub
```

```

' Esta subrotina é chamada quando o utilizador selecionar o botão Sair
Private Sub BtSair_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtSair.Click
    End
    End Sub
End Class

```

### Objeto ComboBox

Arraste um controlo do tipo “*ComboBox*” para a janela “Form1”. A partir desse momento passa a ter um objeto com o nome “*ComboBox1*”. Este controlo permite ao programador criar uma lista de itens/texto e ao utilizador ler os itens e selecionar um deles.

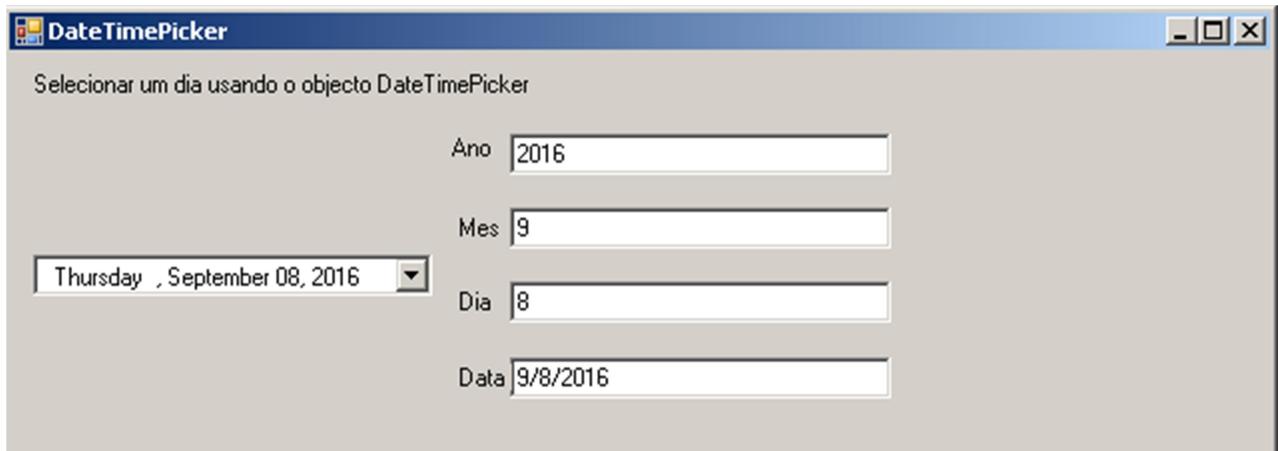
Escreva a mensagem “ <i>ListaItems</i> ” na propriedade text	ComboBox1.text = "Lista Items "
Altere a cor de fundo do botão para amarelo	ComboBox1.BackColor = Color.Yellow
Altere a cor da letra para azul	ComboBox1.ForeColor = Color.Blue
Altere a altura da caixa de texto para 20 pixels	ComboBox1.height = 50
Altere a largura da caixa de texto para 20 pixels	ComboBox1.Width = 100
Devolve o número de items presentes na combobox	ComboBox1.Items.Count
Adiciona um novo Item/linha à combobox	ComboBox1.Items.Add ("A")
Remove um item à combobox	ComboBox1.Items.remove("A")
Remove todos os itens da combobox	ComboBox1.Clear()
Devolve o item que o utilizador selecionou	ComboBox1.SelectedItem
Altere o nome deste objeto para “ <i>Cb_Letras</i> ”	ComboBox1.Name = "CB_Letras"

### **2.3.4 Exemplo “Calendário”**

Neste exemplo pretende-se apresentar o controlo ***DateTimePicker***.

Quando o utilizador selecionar uma nova data, usando o controlo ***DateTimePicker*** (à esquerda na figura), o sub procedimento *DateTimePicker1\_ValueChanged* será executado e nas textbox da direita aparecerá o ano, o mês , o dia e a data que o utilizador selecionou.

Nesta interface o programador utiliza um controlo do tipo ***DateTimePicker***, quatro controlos tipo ***Label***, e quatro controlos do tipo ***Textbox*** .



Public Class Form1

```
' Quando o utilizador selecionar na DateTimePicker uma nova data esta subrotina é chamada
Private Sub DateTimePicker1_ValueChanged(ByVal sender .....
```

```
txtAno.Text = DateTimePicker1.Value.Year
txtMes.Text = DateTimePicker1.Value.Month
txtDia.Text = DateTimePicker1.Value.Day
txtData.Text = DateTimePicker1.Value.Date
```

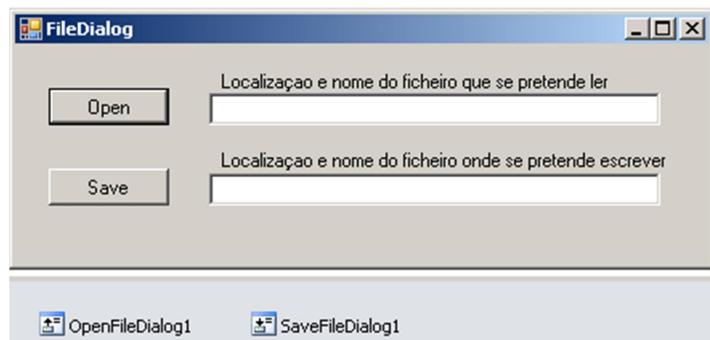
```
End Sub
End Class
```

### 2.3.5 Exemplo “Selecionar um ficheiro no disco”

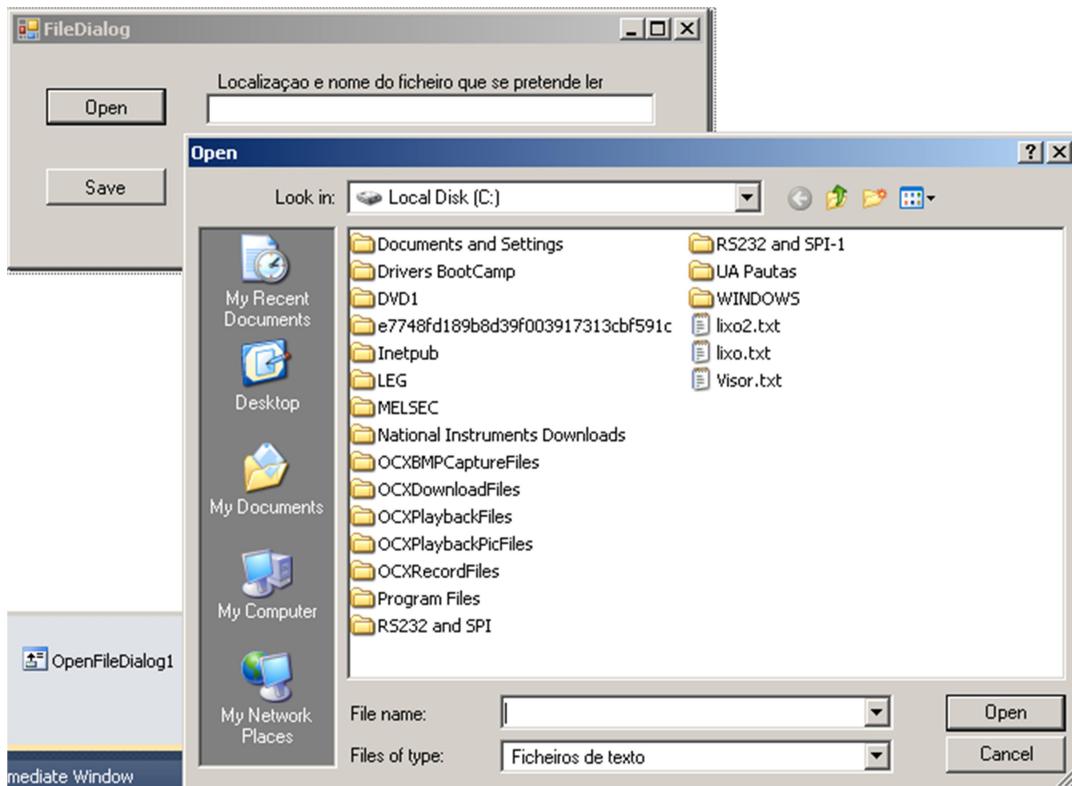
Neste exemplo pretende-se apresentar os controlos **SaveFileDialog** e **OpenFileDialog**.

Ao contrário do que o nome deste tipo de objetos daria a entender, eles não permitem gravar ou abrir ficheiros em disco, permitem apenas que o utilizador veja o conteúdo do disco e selecione o diretório e o nome do ficheiro que prenda posteriormente abrir ou onde pretende gravar dados.

Neste exemplo, o programador utiliza dois controlos do tipo **Label**, dois controlos do tipo **Textbox**, dois controlos do tipo **Button**, um controlo do tipo **SaveFileDialog** e outro do tipo **OpenFileDialog**.



Quando o utilizador pressionar o botão “Open”, o método ShowDialog() do objeto OpenFileDialog é executado e surge no ecrã uma janela de diálogo que permite ao utilizador navegar no disco, visualizar os diretórios e ficheiros existentes.



Quando o utilizador selecionar um dos ficheiros existentes, a sua localização e nome ficam guardados na propriedade **FileName** do objeto **OpenFileDialog1** e essa localização e nome podem ser por exemplo visualizadas numa textbox (TextBox1.Text = OpenFileDialog1.FileName).

Se o utilizador sair da janela de diálogo sem selecionar um ficheiro “Cancel”, a função OpenFileDialog1.ShowDialog() retorna o número dois, o que poderá ser visualizado numa textbox  
( TextBox1.Text = OpenFileDialog1.ShowDialog() ).

Public Class Form1

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
' Filter: Quando a Janela de di-logos aparecer ao utilizador
'serao mostrados apenas os ficheiros com a extensao *.bmp
OpenFileDialog1.Filter = " Image Files (*.bmp;*.jpg)| All files|*.*"
OpenFileDialog1.FileName = ""
```

End Sub

```

Private Sub BtOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtOpen.Click
    Dim ii As Integer

    ' Mostra ao utilizador a janela de diálogo
    ' Na janela de diálogo o utilizador pode retornar "Cancel"
    ' sem selecionar nenhum ficheiro.
    ' Se o ficheiro foi seleccionado a função retorna 1 caso contrário retorna 2
    ii = OpenFileDialog1.ShowDialog() ' ii=2 - cancel ii = 1 - open

    ' Quando o utilizador sai da janela de diálogo,
    ' o nome e a localização do ficheiro seleccionado
    ' está memorizado na propriedade "FileName"
    TextBox1.Text = OpenFileDialog1.FileName

    ' Se um ficheiro imagem foi seleccionado
    If ii <> 2 Then
        PictureBox1.Image = Image.FromFile(OpenFileDialog1.FileName)
    End If

End Sub

```

### Objeto OpenFileDialog

Arraste um controlo do tipo “*OpenFileDialog*” para a janela “Form1”. A partir desse momento passa a ter um objeto com o nome “*OpenFileDialog1*”. Este controlo permite ao programador navegar no disco e visualizar os diretórios e ficheiros existentes. Depois da Janela de diálogo estar aberta *OpenFileDialog1.ShowDialog()* o utilizador pode selecionar um ficheiro ou cancelar e sair da janela. Se sair da janela depois de ter selecionado um ficheiro esta função retorna o número 1 e o nome e a localização do ficheiro é guardada na propriedade *FileName*, se o utilizador sair da janela de diálogo fazendo “cancel” esta função retorna o número 2.

---

Quando a janela de diálogo ficar visível apenas mostra os ficheiros com a(s) extensões indicadas	<i>OpenFileDialog1.Filter = "bmp *.bmp   jpg *.jpg"</i>
Mostra a janela de diálogo ao utilizador e retorna o número 1 ou o número 2 quando o utilizar sair da janela	<i>TextBox1.Text = OpenFileDialog1.ShowDialog()</i>
Depois do utilizador ter seleccionado um ficheiro e saído da janela, contém o nome e localização do ficheiro seleccionado.	<i>textBox1.Text = OpenFileDialog1.FileName</i>

---

### **2.3.6 Exemplo “Ler e gravar ficheiros de texto”**

Neste exemplo pretende-se abrir ficheiros de texto e visualizar o seu conteúdo numa caixa de texto. Pretende-se também gravar uma string num ficheiro texto.

## *StreamWriter e StreamReader*

---

Os objetos do tipo **StreamReader** e **StreamWriter** permitem, respectivamente, ler ou escrever em ficheiros de texto.

Para poder usar objetos do tipo StreamReader e StreamWriter deve importar o “System.IO”, acrescentando no seu código:

```
Imports System.IO
```

**Para escrever** o texto “Ola” no ficheiro “c:\exemplo.txt”, deve declarar um objeto do tipo StreamWriter, por exemplo com o nome “**ficheiroguardar**”, fazendo:

```
Dim ficheiroguardar as StreamWriter
```

Para guardar o texto “Ola” no ficheiro “c:\exemplo.txt”, apagando o que lá estava antes, deve fazer:

```
ficheiroguardar = new StreamWriter (“c:\exemplo.txt”, False )  
ficheiroguardar.write(“Ola”)
```

Para acrescentar o texto “Ola”, **mantendo** o que estava antes, deve fazer:

```
ficheiroguardar = new StreamWriter (“c:\exemplo.txt”, True )  
ficheiroguardar.write(“Ola”)
```

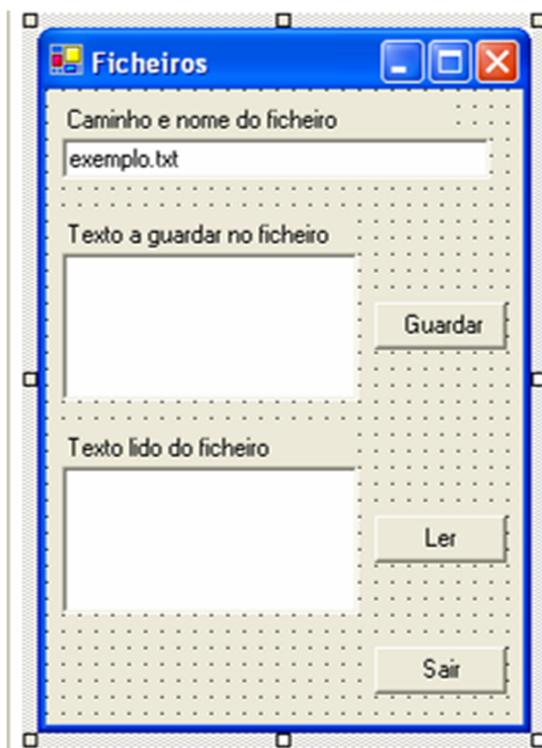
Finalmente, para que o texto seja de facto escrito em disco deve fazer:

```
ficheiroguardar.close()
```

**Para ler** o conteúdo de um ficheiro de texto, por exemplo, o conteúdo do ficheiro “c:\exemplo.txt”, deve declarar um objeto do tipo StreamReader, por exemplo com o nome “**ficheiroler**”, fazendo:

```
Dim ficheiroler as StreamReader  
ficheiroler = new StreamReader(“c:\exemplo.txt”)  
Textbox1.text = ficheiroler.ReadToEnd()  
ficheiroler.Close()
```

Após esta apresentação dos objetos do tipo StreamReader e StreamWriter será fácil perceber o exemplo seguinte:



```
Namespace System.IO
    'necessárias para a manipulação de ficheiros
    Imports System.IO

    Public Class Form1
        Inherits System.Windows.Forms.Form

        Windows Form Designer generated code

        Private Sub btnGuardar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGuardar.Click
            Dim ficheiroGuardar As StreamWriter = New StreamWriter(txtPath.Text, False)
            ficheiroGuardar.Write(txtDadosGuardar.Text)
            ficheiroGuardar.Close()
        End Sub

        Private Sub btnAbrir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAbrir.Click
            Dim ficheiroLer As StreamReader = New StreamReader(txtPath.Text)
            txtDadosLer.Text = ficheiroLer.ReadToEnd()
            ficheiroLer.Close()
        End Sub

        Private Sub btnSair_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSair.Click
            End
        End Sub
    End Class
```

## 2.4 Linguagem de programação

Mesmo depois da interface gráfica de um novo programa estar concebida e implementada o nosso programa tem de ser capaz de processar os dados recebidos através dessa interface.

Tal como todas as linguagens de programação o VBasic permite ao programador criar variáveis, usar operações matemáticas, operações booleanas, instruções de controlo de fluxo e também agrupar o seu código em funções ou procedimentos.

### **2.4.1 Bases numéricicas, prefixos**

Em VBasic, o programador tem a liberdade de escrever os números em várias bases numéricicas: Decimal, Octal, Hexadecimal mas tem de indicar ao compilador em qual das bases escreve um determinado número. Para isso tem de colocar um prefixo **&H** ou **&O** antes do número

Hexadecimal	<b>&amp;H9f</b>	' O número hexadecimal 9f corresponde ao número decimal 159
Octal	<b>&amp;O65</b>	' O número octal 65 corresponde ao número decimal 53

Por defeito, se nada for dito ao compilador, ele assume que o número foi escrito na base decimal.

Mas, mesmo na base decimal um número pode ser positivo ou negativo, pode ser pequeno ou grande e necessitar de vários bytes em memória para o armazenar, pode também ser inteiro ou real. Em VBasic, o programador pode indicar ao compilador de que forma um número deve ser considerado escrevendo a seguir ao número as letras correspondentes ao tipo de número. Por exemplo, o programador pode querer que o número 100, escrito na base decimal, seja tratado como um número inteiro ou como um número real (com vírgula):

Números inteiros		
<i>Inteiro sem sinal</i>	<b>100UI</b>	'Unsigned Int
<i>Inteiro curto</i>	<b>100S</b>	'Short
Números reais		
<i>Single</i>	<b>100F</b>	'Float
<i>Double</i>	<b>100R</b>	'Real

### **2.4.2 Variáveis**

Para o programa poder guardar na memória RAM do computador: números, texto e dados em geral o programador pode declarar diferentes tipos de variáveis. Estas variáveis podem ser escalares, matriciais ou estruturas complexas.

Byte, Integer, Decimal, Long, Boolean, String, Variant

Em VBasic, o programador tem vários tipos de variáveis à disposição, dependendo do tipo de valores que pretender guardar pode usar variáveis do tipo “*char*”, “*integer*” e outros tipos.

Uma variável do tipo “*Byte*” ocupa 8 bits na memória do computador. Por isso, estas variáveis podem conter números positivos de 0 a 255 ou, números negativos e positivos de -128 a 127 “*Signed Byte - SByte*”.

O bit mais significativo das variáveis pode ser usado para representar o sinal (-). Quando este bit vale “1” o número é negativo. Quando as variáveis são do tipo “*unsigned*”, sem sinal, todos os bits, mesmo o mais significativo é usado para representar um número positivo.

Uma variável do tipo “*char*” ocupa 32 bits na memória do computador. Mas só pode conter valores correspondentes a caracteres da tabela ascii.

Uma variável do tipo “*Short*” ocupa 16 bits na memória do computador. Por isso, estas variáveis podem conter números negativos e positivos de -32768 a 32767 ou, números positivos de 0 a 65535 “*Unsigned Short - UShort*”.

Uma variável do tipo “*integer*” ocupa 32 bits na memória do computador. Por isso pode conter números positivos de 0 a  $2^{32}$  “*unsigned integer - UInteger*” ou, números negativos e positivos “*Integer*”.

As variáveis do tipo “vírgula flutuante”: *Single* e *Double* ocupam 4 e 8 bytes respectivamente na RAM.

As variáveis, mais não são que posições da memória RAM do computador, onde podemos escrever ou ler valores. Esses valores podem ser numéricos ou alfanuméricos. Se forem alfanuméricos é necessário codificá-los em números de acordo, por exemplo, com a norma American Standard Coded Information Interchange – ASCII.

### *Grandezas escalares*

Para declarar uma variável em VBasic chamada *Contador* do tipo *Integer* deve fazer:

```
Dim Contador as Integer
```

Para atribuir um valor a esta variável, por exemplo o valor 100, pode fazê-lo logo no momento da sua declaração:

```
Dim Contador as Integer = 100
```

' Números inteiros com sinal

```
Dim s As Short = &H7FFF
```

' 2 bytes ( -32 768 a +32 767 )

```
Dim i16 As Int16 = &H7FFF
```

' 2 bytes ( -32 768 a +32 767 )

```
Dim i32 As Int32 = &H7FFFFFFF
```

' 4 bytes ( - 2 147 483 647 a + 2 147 483 647 )

```

Dim i As Integer = &H7FFFFFFF      ' 4 bytes (-2 147 483 647 a + 2 147 483 647)
Dim i64 As Int64 = &H7FFFFFFFFFFFFFFF   ' 8 bytes
Dim il As Long = &H7FFFFFFFFFFFFFFF    ' 8 bytes

```

#### ' Numeros inteiros sem sinal

```

Dim b As Byte = &HFF          ' 1 byte (0 a 255)
Dim us As UShort = &HFFF        ' 2 bytes (0 a 65 535)
Dim ui16 As UInt16 = &HFFFF       ' 2 bytes (0 a 65 535)
Dim ui32 As UInt32 = &H7FFFFFFF     ' 4 bytes (0 a 2 147 483 647)
Dim ui AsUInteger = &H7FFFFFFF      ' 4 bytes (0 a 2 147 483 647)
Dim ui64 As UInt64 = &H7FFFFFFFFFFFFFFF  ' 8 bytes
Dim ul As ULONG = &H7FFFFFFFFFFFFFFF   ' 8 bytes

```

#### ' Numeros reais

```

Dim de As Decimal = 3.123456789
Dim si As Single = 3.123456789
Dim d As Double = 3.123456789

```

#### 'Texto

```
Dim st As String = "Ola"
```

## Arrays e Strings

Uma matriz não é mais que um conjunto de posições da RAM do computador. As matrizes podem ser unidimensionais "vectores" ou multidimensionais.

Uma “*string*” é um vector que ocupa várias posições de memória, onde cada caracter é guardado numa posição de memória do vector e o último elemento do vector tem todos os bits a zero. Cada caracter da “string” é codificado em binário, de acordo com a tabela ASCII, e só depois é guardado na memória do computador.

Dim letra(9) as char	' Vector com 10 elementos: letra(0) a letra(9)		
Letra(0) = "J"	' J	= 74 <sub>dec</sub> = &H4A	= 0b 01001010
Letra(1) = "o"	' o	= 111 <sub>dec</sub> = &H6F	= 0b 01101111
Letra(2) = "s"	' s	= 115 <sub>dec</sub> = &H73	= 0b 01110011
Letra(3) = "e"	' e	= 101 <sub>dec</sub> = &H65	= 0b 01100101
Letra(4) = chr(0)	' valor nulo = 0	= &H00	= 0b 00000000
Textbox1.text = Letra	' na textbox aparece a palavra Jose		

```
Dim nome(10)(50) as char      ' Matriz de duas dimensões
```

#### ' Variáveis globais

```

Dim cc(.) as char = {{"J","o"}, {"s,"e"}}
Dim nome() as char = {"j","o","s",chr(101)}
Dim aluno() as char ="Jose"
Dim I as short = &H4A

```

```
Private sub Form1_Load( ....
```

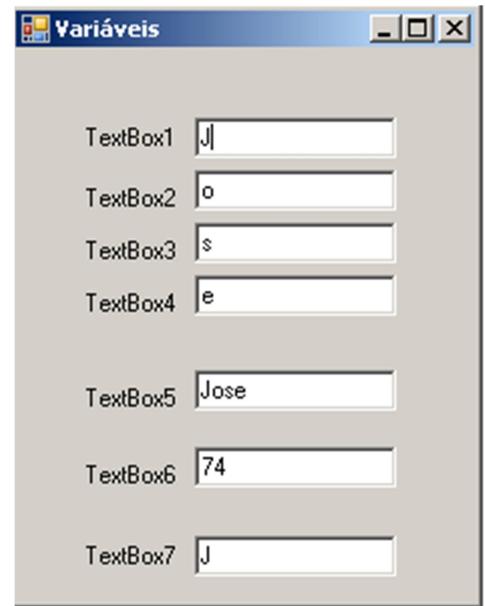
```

TextBox1.Text = cc(0,0) ' J
TextBox2.Text = cc(0,1) ' o
TextBox3.Text = cc(1,0) ' s
TextBox4.Text = cc(1,1) ' e

TextBox5.Text = nome ' Jose
TextBox6.Text = I ' 74
TextBox7.Text = chr(I) ' J

end sub

```



### 2.4.3 Declaração de Funções e Procedimentos

As funções e os procedimentos permitem agrupar linhas de código numa única entidade que pode ser chamada a partir de outros locais do programa, evitando repetir as mesmas linhas de código ao longo do programa.

O exemplo seguinte descreve como se pode declarar e usar um procedimento. Neste exemplo a função tem o nome *Mensagem()*.

Apesar do procedimento *Mensagem()* ser o primeiro a ser declarado neste exemplo, ele é chamado depois e a partir do sub procedimento *Form\_Load(...)*. Neste exemplo o procedimento não recebe nem retorna valores.

```

Sub Mensagem ()
    TextBox1.Text = "Mensagem"           ' Texto
End Sub

Sub Form_load (...)
    Mensagem ()
End Sub

```

Existe uma diferença entre *funções* e *procedimentos*. Os dois permitem “agrupar” código, tornando mais fácil a sua organização e reutilização ao longo do programa, mas só as funções podem retornar valores.

No exemplo seguinte é declarada uma função, intitulada *Pi()*. Esta função retorna o número 3,14 quando é chamada no sub procedimento *Form\_Load(...)*.

```

Function Pi() Double           ' Esta função retorna um número real do tipo
Double                                         Double
    Return 3.14                         ' Retorna o número 3.14
End Function

```

```

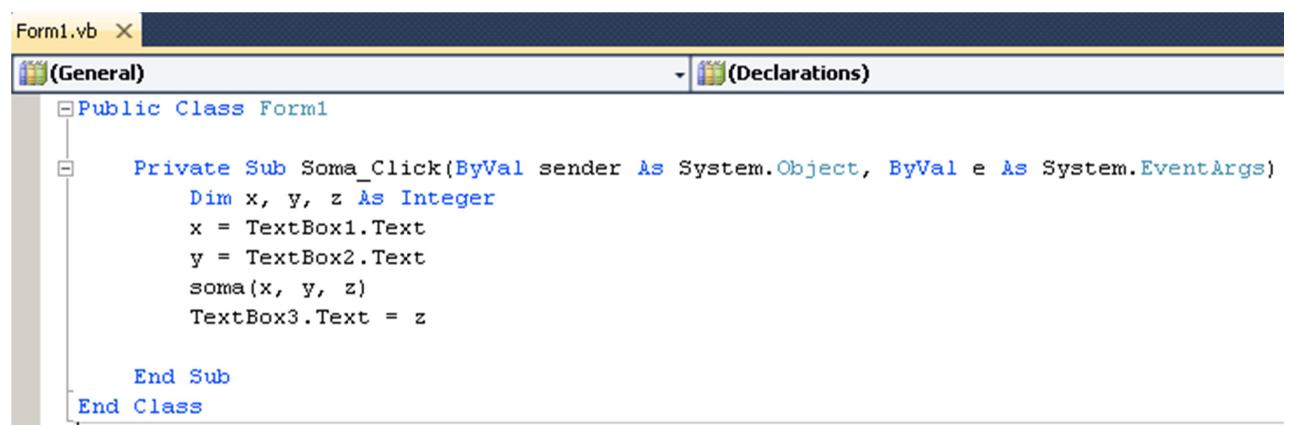
Private sub Form_load ( ... )
    TextBox1.Text = Pi()
    ' Na caixa de texto aparece o número 3.14
End Sub

```

### Funções - passagem de parâmetros por valor ou por referência

Tanto as funções como os procedimentos podem receber parâmetros.

Neste exemplo, a função **soma(X,Y,Z)** recebe os parâmetros **X** e **Y** por valor. Ou seja, o valor/conteúdo de **X** e **Y** são passados para os parâmetros internos “**a**” e “**b**” da função **soma()**. O parâmetro **a** recebe o valor de **X** e o parâmetro **b** recebe o valor de **Y**. Mas as variáveis **X**, **Y** e os parâmetros **a** e **b** correspondem a zonas de memória diferentes.



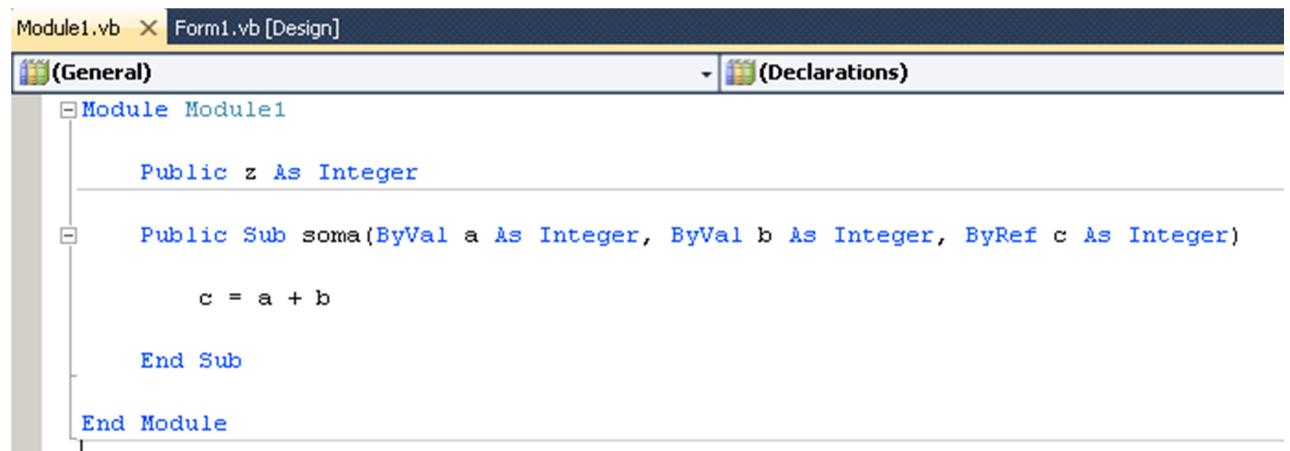
```

Form1.vb x
General (Declarations)
Public Class Form1
    Private Sub Soma_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim x, y, z As Integer
        x = TextBox1.Text
        y = TextBox2.Text
        soma(x, y, z)
        TextBox3.Text = z
    End Sub
End Class

```

Pode-se observar na figura seguinte que a função **soma** recebe por valor **ByVal** dois parâmetros do tipo inteiro **a** e **b**. Depois de efectuada a soma esse valor é atribuído ao parâmetro **c** que é passado por referência **ByRef**.

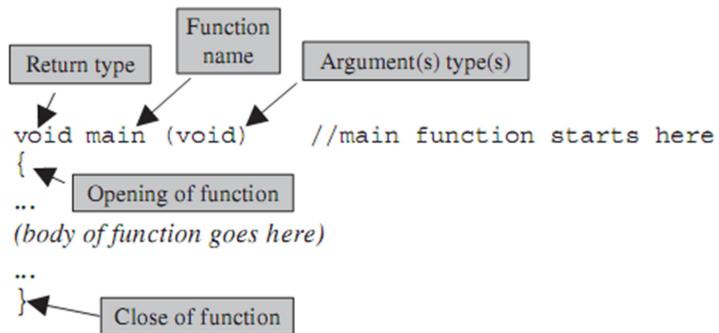
Como o parâmetro **c** é passado por referência, tanto o parametro **c** como a variável **Z** correspondem à mesma zona de memória. Ou seja, atribuir valores a **c** dentro da função **soma** é o mesmo que atribuir esses valores à variável **Z** fora da função **soma()**.



```

Module1.vb x Form1.vb [Design]
General (Declarations)
Module Module1
    Public z As Integer
    Public Sub soma(ByVal a As Integer, ByVal b As Integer, ByRef c As Integer)
        c = a + b
    End Sub
End Module

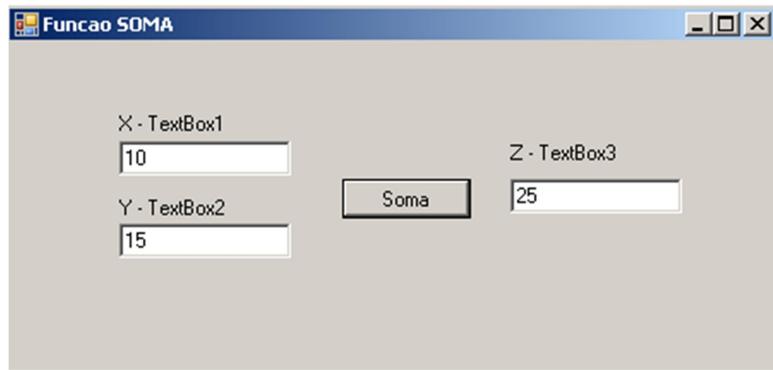
```



### Funções externas e Variáveis externas

Se o seu programa, em “VBBasic”, estiver organizado em vários ficheiros de texto, pode declarar as variáveis ou as funções num deles e utilizá-las noutras ficheiros, mas tem de declarar as variáveis como sendo do tipo “*public*” e devem ser declaradas fora de qualquer função.

No exemplo seguinte pretende-se somar os dois valores. O utilizador introduz dois valores nas caixas de texto da esquerda (TextBox1 e TextBox2), e o resultado da soma é apresentado na caixa de texto da direita (TextBox3).



Foram declaradas três variáveis inteiros **X,Y,Z** e uma função **soma()** no ficheiro module1.vb que podem ser usadas noutras ficheiros, nomeadamente no ficheiro form1.vb

Quando o utilizador do programa selecionar o botão “Soma”, o sub procedimento **Soma\_Click** é executado. Nessa altura, de acordo com o código escrito neste exemplo, o conteúdo das caixas de texto da esquerda (TextBox1 e TextBox2) é atribuído às variáveis globais **X** e **Y** e é chamada a função global **soma()** do module1.vb. Na função **soma()**, é atribuída à variável global **Z** o resultado da soma. Quando a execução do programa retorna ao sub procedimento **Soma\_Click(...)** é atribuído à caixa de texto da direita (TextBox3) o valor da variável global **Z**.

```

Module1.vb X Form1.vb [Design] X Form1.vb X Solution Explorer X
Module1 (Declarations)
Module Module1
    Public x, y, z As Integer
    Public Sub soma()
        z = x + y
    End Sub
End Module

Form1.vb (General) (Declarations)
Public Class Form1
    Private Sub Soma_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Soma.Click
        x = TextBox1.Text
        y = TextBox2.Text
        soma()
        TextBox3.Text = z
    End Sub
End Class

WindowsApplication1
My Project
Form1.vb
Module1.vb

```

#### 2.4.4 Tipos de operadores

##### Operadores elementares (= + - \* /)

Em “VBasic” os operadores de atribuição “=”, soma “+”, subtração “-”, multiplicação “\*” e divisão “/” têm a sintaxe que o exemplo seguinte apresenta:

```

Private Sub main()
    Dim X,Y,Z as Char
    X= 3
    Y= 2
    Z= X+Y           // Depois desta instrução a variável Z contém o valor 5
    Z= Z * 2 + 1     // Depois desta instrução a variável Z contém o valor 11
    Z= Z / 2          // Depois desta instrução a variável Z contém o valor 5
    TextBox1.Text =Z   // Depois desta instrução, a caixa de texto apresenta o valor 5
End Sub

```

##### Operadores relacionais

Os operadores booleanos comparam dois valores e com base na comparação realizada retornam um valor booleano “Verdadeiro” ou “Falso”.

Maior que	>
Maior ou igual	>=
Menor que	<
Menor ou igual	<=
Igual	=
Diferente	<>

```

Public sub main()
    Dim A,B,C as Integer
    A=10
    B=15
    C=20

    If A < B Then    TextBox1.Text = " O resultado da comparação é TRUE, 10 é menor que
                      15"

```

```

If B > A Then    TextBox1.Text = "O resultado da comparação é TRUE, 15 é maior que 10"
If A <> B Then  TextBox1.Text = "O resultado da comparação é TRUE, 10 é diferente de
10"
If A = A Then    TextBox1.Text = "O resultado da comparação é TRUE, 10 é igual a 10"
EndSub

```

## Operadores lógicos

Intercepção de conjuntos	<b>And</b>
Reunião de conjuntos	<b>Or</b>
Negação	<b>Not</b>
Ou exclusivo	<b>Xor</b>

A	B	A And B	A Or B	Not A	A Xor B
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

*Private Sub main()*

```

Dim A,B as Boolean
A=True
B=False

```

```

If B Or A Then      TextBox1.Text = "True ou False é True"
If A Xor B Then    TextBox1.Text = "True xor False é True"
If Not B Then       TextBox1.Text = "Negação de False é True "
If Not (A And B) Then TextBox1.Text = "A negação de (True and False) = True"

```

*EndSub*

## **2.4.5 Estruturas de controlo**

A execução de um programa é sequencial, cada instrução é gravada na memória do computador em posições consecutivas e é executada pela ordem com que foi gravada, a menos que o programador tenha inserido no próprio código instruções que permitam alterar essa sequência de execução.

### O ciclo While ... End While

A instrução WHILE(*condição*) ... EndWhile permite repetir a execução de algumas linhas código, mais exatamente das linhas de código que tenham sido escritas entre a palavra While e a palavra EndWhile. Estas linhas de código são repetidas enquanto a *condição* booleana do WHILE for verdadeira.

A *condição* do ciclo WHILE pode conter vários operandos e operadores mas o resultado final das operações tem de ser um valor booleano, verdadeiro ou falso. Enquanto a condição for verdadeira o ciclo repete-se.

<b>While</b> ( <i>condição</i> )	i = 0	Depois de 3 ciclos, TextBox1.Text = 3
<i>instruções</i>	while (i < 3)	
<i>instruções</i>	i=i+1;	

**EndWhile**

TextBox1.Text = i

### O ciclo For ... Next

Este ciclo permite repetir a execução das linhas de código que estiverem escritas entre o **For** e o **Next** do próprio ciclo. Este ciclo é semelhante ao ciclo WHILE, contudo a “condição” do ciclo WHILE pode assumir um valor falso em função da execução das instruções do próprio ciclo, fazendo com que o ciclo possa terminar mais cedo ou tarde, ao passo que o ciclo FOR tem um número de repetições predefinido.

<b>for</b> i=0 <b>to</b> n <b>step</b> k	for i=0 to 10 step 1	
<i>instruções</i>	...	
<i>instruções</i>	Next	

Next

TextBox1.Text= i      ‘Depois de 10 ciclos, TextBox1.Text = 11’

### Exemplo 1

Neste exemplo todos os 16 elementos da matriz Galo(3,3) ficam com os valores indicados na tabela. Os dois ciclos **For** do exemplo repetem-se 4 vezes. Contudo, como o segundo ciclo For foi declarado dentro do primeiro, por cada ciclo do primeiro (**For i**) o segundo ciclo (**For j**) repete-se 4 vezes. Ou seja, a instrução  $Galo(i,j) = i * j$  repete-se 16 vezes.

Public Galo(3,3) as integer  
Public i , j as integer

For i= 0 to 3 Step 1  
    For j= 0 to 3 Step 1  
        Galo( i , j ) = i \* j  
        TextBox1.Text = Cstr(i) + Cstr(j)  
    Next  
Next

Na Matriz Galo(i,j) o índice “j” corresponde às colunas e o índice “i” às linhas.

Galo(i,j)	j=0	1	2	3
i=0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	9

### If ... End If

O programador pode usar a instrução IF quando se pretende executar algumas linhas de código se, e apenas se “**IF**”, uma determinada condição for verdadeira, ou pelo contrário “**Else**”, executar outras linhas de código se a condição for falsa.

<b>if</b> ( <i>condição</i> )	i= 5;	Como i = 5, a condição (i<10) é verdadeira
‘se condição verdadeira’	<b>if</b> (i < 10 )	TextBox1.Text = 3 Se i fosse maior que 10 a condição (i<10) era falsa

<pre> instruções; else 'se condição falsa instruções; End If </pre>	<pre> 'Se verdadeira      TextBox1.Text = 2 TextBox1.Text = 3 else 'Se falsa TextBox1.Text = 2 End If </pre>
---	--

Em suma, esta instrução permite que apenas um de dois conjuntos de instruções seja executado, em função de uma *condição* ser verdadeira ou falsa.

### Select Case .... End Select

A instrução **Select case** permite que apenas um de muitos conjuntos de instruções seja executado, Consoante o valor o valor numérico ou alfanumérico de uma variável será executado apenas o caso correspondente (**case**).

<pre> Dim i as Integer i = 1 Select Case i Case 1: instruções Case 2: instruções Case Else: instruções End Select </pre>	<pre> Dim s as String s = "2" Select Case (s) Case "1": TextBox1.Text = "Um" Case "2": TextBox1.Text = "Dois" Case Else: TextBox1.Text = "Outro" End Select </pre>	<pre> Como s = "2,"           TextBox1.Text = "Dois" TextBox1.Text = "Dois"   Se s = "1" Se s = "1"              TextBox1.Text = "Um" </pre>
--	--	--

### **2.4.6 Biblioteca de funções para manipulação de texto**

#### A função “Mid”

Permite extrair uma *substring* a partir de uma *string* original.

Esta função recebe três parâmetros: uma variável do tipo *string*, contendo um conjunto de caracteres; um número com a posição do primeiro caracter a ler; e o número de caracteres consecutivos a ler.

Mid-Exemplo:

```

Dim strOriginal As String = "0123456789"
MessageBox.Show(Mid(strOriginal, 1, 2))  'devolve o valor "01"
MessageBox.Show(Mid(strOriginal, 5, 3))  'devolve o valor "456"

```

#### A função “Split”

Divide uma *string* em *substrings*.

Esta função recebe uma *string*, um carácter delimitador e devolve um array de *strings*. Em cada posição do array devolvido vem uma parte da *string* original. Por exemplo, se a *string* “String\_original” possuir um espaço em branco entre cada letra e o

utilizador quiser separar as letras, obtendo em cada posição do array uma só letra, pode indicar à função “split” para usar o carácter “espaço” como divisor da *string* original. A este carácter, usado para dividir a “*String\_original*”, chama-se delimitador.

#### Split-Exemplo:

```
Dim strOriginal As String = "o l a"  
Dim arrayDeStrings() As String  
arrayDeStrings = Split(strOriginal, " ")  
MessageBox.Show(arrayDeStrings(0))  
MessageBox.Show(arrayDeStrings(1))  
MessageBox.Show(arrayDeStrings(2))
```

#### A função “InStr”

Permite localizar a posição de uma *substring* numa *string* maior.

Esta função recebe duas *strings*: a *string* maior e a *substring* e devolve um inteiro com posição da *substring* na *string* maior. O valor devolvido pode ser (0) se a *substring* não for encontrada na *string* maior, ou ( $\geq 1$ )

#### InStr-Exemplo:

```
Dim strOriginal As String = "abcdefghijklm"  
Dim strChave As String = "cd"  
MessageBox.Show(InStr(strOriginal, strChave)) 'devolve o valor 3
```

#### As funções “Right” e “Left”

Devolve uma *substring* que contém um determinado número de caracteres, a contar da direita ou da esquerda, da *string* original.

Por existir uma propriedade do form com o mesmo nome é necessário utilizar o caminho completo *Microsoft.VisualBasic.Left* e *Microsoft.VisualBasic.Right*.

#### Right e Left-Exemplo:

```
Dim strOriginal As String = "abcdefghijklm"  
MessageBox.Show(Microsoft.VisualBasic.Left(strOriginal, 3))      'devolve o valor "abc"  
MessageBox.Show(Microsoft.VisualBasic.Right(strOriginal, 2))    'devolve o valor "hi"
```

#### A função “Len”

Devolve um inteiro com o número de caracteres que existem numa *string*.

Esta função recebe uma *string*, da qual queremos saber o seu comprimento, ou seja o número de caracteres que a constituem e devolve um número.

#### Len-Exemplo:

```
Dim strOriginal As String = "abcdefghijklm"  
MessageBox.Show(Len(strOriginal))  'devolve o valor 9
```

### A função “StrComp”

Compara duas *strings*.

Esta função recebe duas *strings* e devolve um valor inteiro, igual a zero se as duas *strings* forem iguais.

StrComp-Exemplo:

```
Dim palavra1 As String = "123"  
Dim palavra2 As String = "123"  
MessageBox.Show(StrComp(palavra1, palavra2)) 'devolve o valor 0
```

### A função “CStr”

Permite converter um número numa *string*. A *string* obtida pode então ser atribuída a variáveis do tipo *string* ou usada em funções de manipulação de *strings*.

CStr-Exemplo:

```
Dim texto As String  
texto = CStr(123)
```

### A função “Chr”

Converte um valor numérico no seu carácter correspondente, de acordo com a tabela ASCII

Chr-Exemplo:

```
MessageBox.Show(Chr(65)) 'devolve o valor "A"
```

### A função “Asc”

Converte um carácter no seu valor decimal de acordo com a tabela ASCII.

Esta função recebe uma string e devolve um valor inteiro.

Asc-Exemplo:

```
MsgBox(Asc("A")) 'devolve o valor 65
```

### A função “Trim”

Rerira os espaços em branco no inicio e no fim de uma string, mas deixa ficar os espaços em branco no meio da string.

Esta função recebe uma string e devolve uma string sem espaços em branco no inicio e no fim.

Trim-Exemplo:

```
Dim strOriginal As String = " ola bom dia "  
MsgBox(Trim(strOriginal)) 'devolve a string "ola bom dia"
```

Veja também as funções:  
InStrRev; UCASE(Text1); LCASE(Text1); LTrim, RTrim.

### 2.4.7 Funções matemáticas

$^$	- Expoente
Sqr	- Raiz quadrada
Mod	- Resto da divisão inteira ( Res = I Mod 10 )
\	- Resultado inteiro da divisão
Abs(n)	- Valor absoluto de n;
Atan(n)	- Arco Tangente de n, n em radianos;
Cos(n)	- Coseno de n, n em radianos;
Sin(n)	- Seno de n, n em radianos;
Tan(n)	- Tangente de n, n em radianos;
Exp(n)	- Expoente de n;
Sign(n)	- Devolve o sinal de um numero, -1 se $<0$ , 1 se $>0$ ;

### 2.5 Projeto com vários Forms e Módulos

---

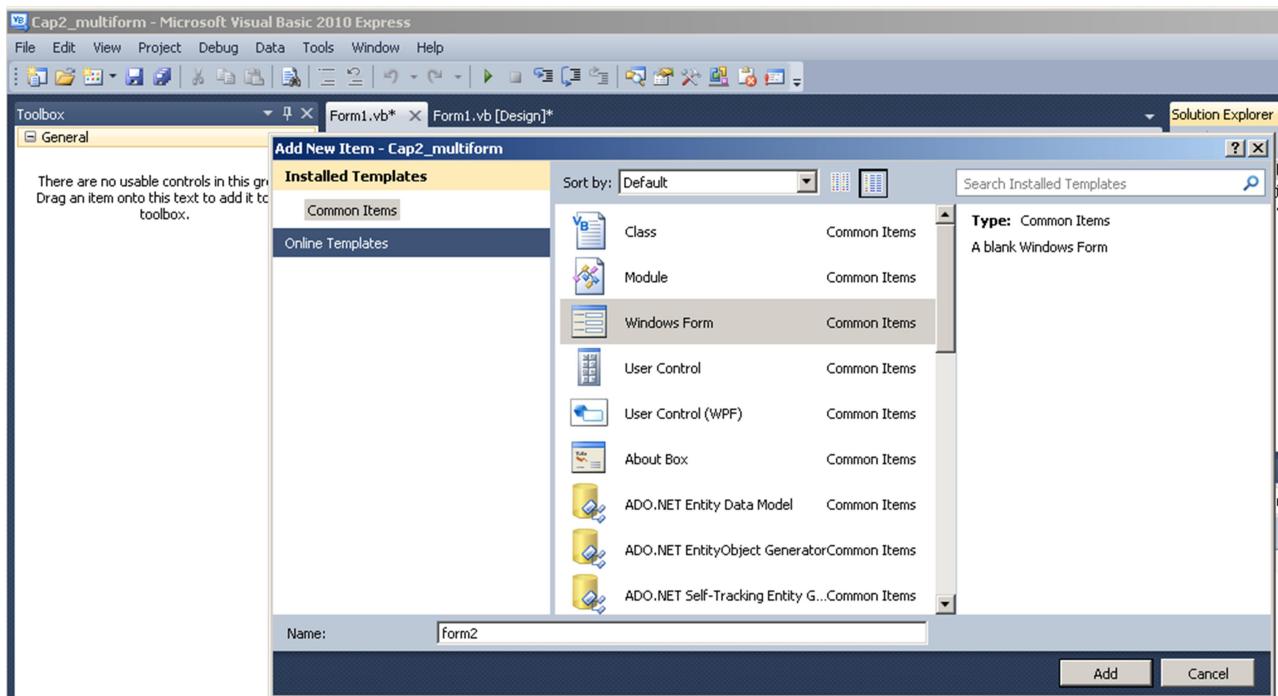
Nesta secção vamos abordar o projectos em VBasic com várias interfaces (forms), módulos, variáveis e funções globais.

Como até agora apenas usámos um Form nos projetos, sempre que o programa era executado era esse “form” que aparecia no ecrã.

#### 2.5.1 Criar um segundo Form

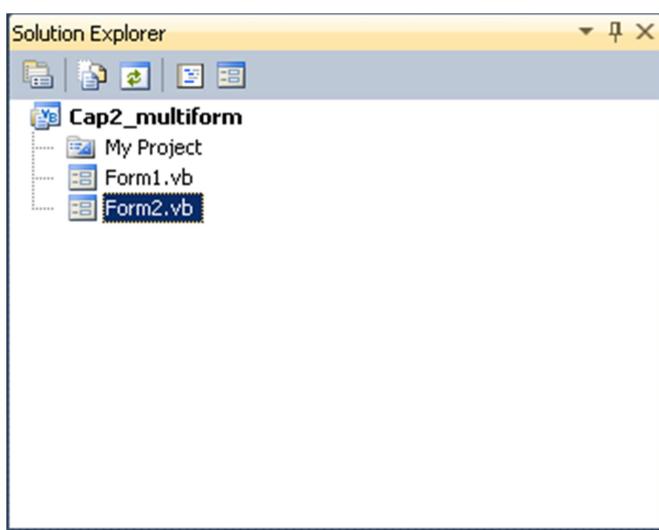
Vamos criar um segundo “*Form*” com o nome “*Form2*” onde o utilizador poderá introduzir texto numa TextBox. Pretende –se que ao voltar ao “*Form*” inicial apareça numa TextBox o nome que o utilizador introduziu na segundo *Form*.

Para criar um segundo “**Form**” selecione a opção “**Add Windows form**” no menu “Project”. Crie um *Form* com o nome **Form2**.



Nessa altura poderá observar na janela “**Solution Explorer**” que o seu projeto passou a ter dois forms: o *Form1* e o *Form2*.

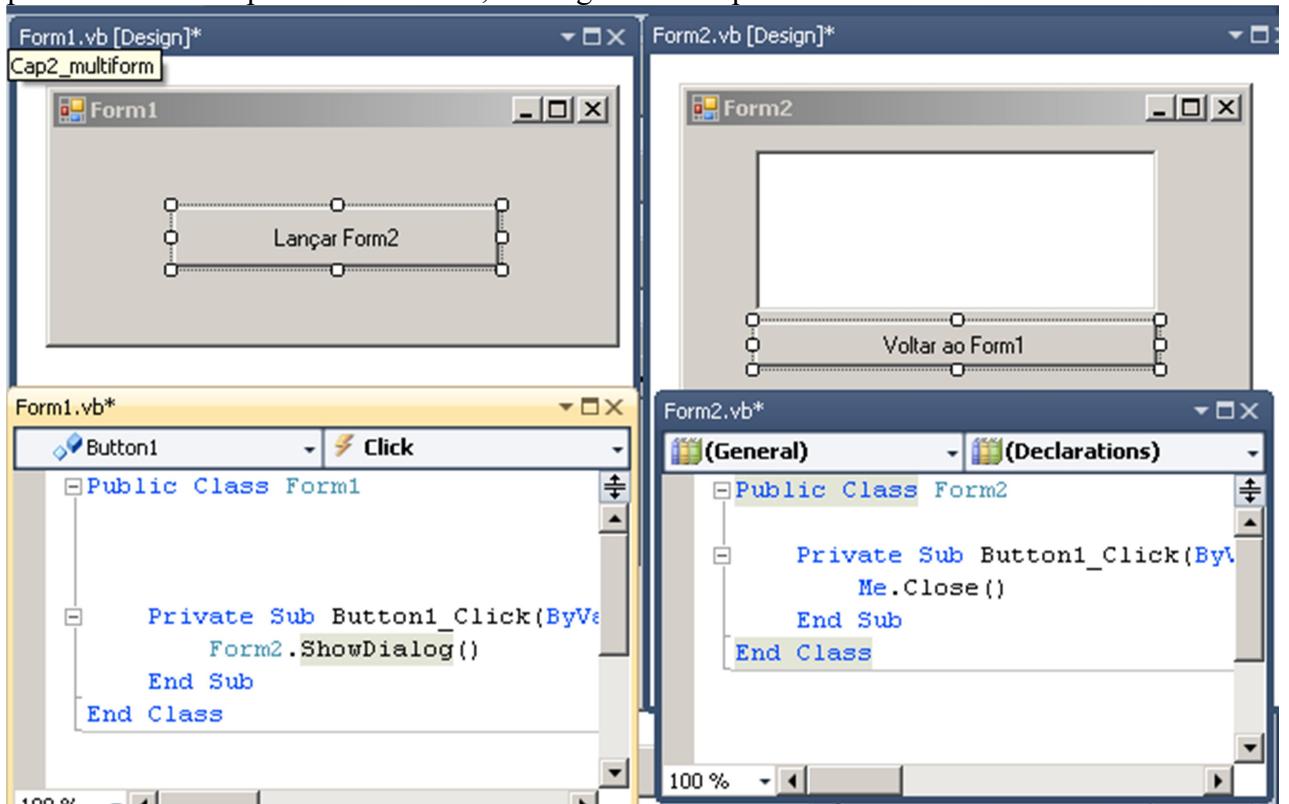
Cada um deles possui uma janela de interface (Form1.vb design e Form2.vb design) e um ficheiro de código (Form1.vb e Form2.vb).



## 2.5.2 Alternar entre Forms

Se no *Form1* usar a função **Form2.ShowDialog()** ativa e visualiza o *Form2*. Para fechar o *Form2*, num dos subprocedimentos do *Form2* deve fazer **me.Close()**.

O exemplo seguinte possui dois *Forms*, cada um com um botão, que quando for pressionado muda para o outro *Form*, o código é auto explicativo.



## 2.5.3 Passar dados entre Forms

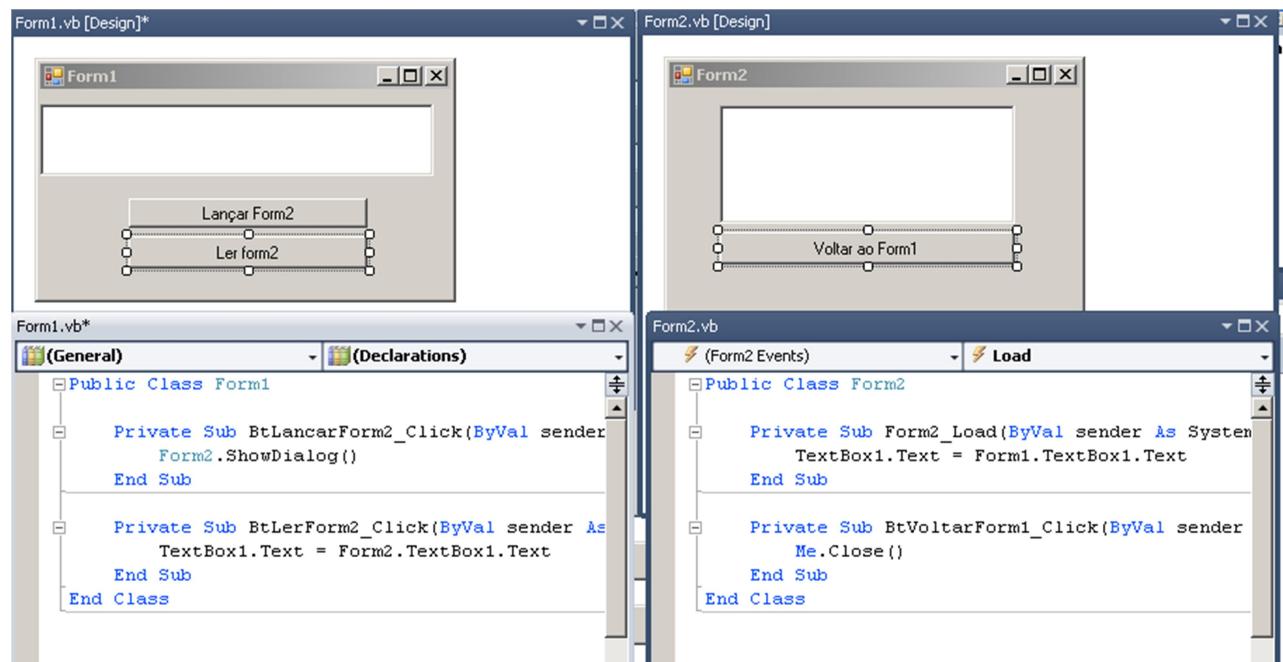
Em cada *Form* o utilizador pode introduzir dados, mas pode ser necessário que os dados introduzidos num *Form* tenham de ser tratados noutro *Form*. Poem-se por isso a necessidade de passar dados entre *Form*.

Neste exemplo, o utilizador introduziu texto na “TextBox1” do *Form2* e quer ver esse texto no objeto “TextBox1” do *Form1*. De facto, existem dois objetos “TextBox1”, um pertence ao *Form1* e o outro ao *Form2*.

O *Form1* tem dois botões, um permite ativar o *Form2* e o outro botão permite visualizar na TextBox1 do *Form1* o conteúdo do segundo “**Form2.TextBox1**”.

Para identificarmos objetos ou variáveis de um *Form*, a partir de outro *Form*, devemos referir o nome do *Form* onde essas variáveis ou objetos foram declarados, por exemplo:

***TextBox1.text = Form2.TextBox1.text***

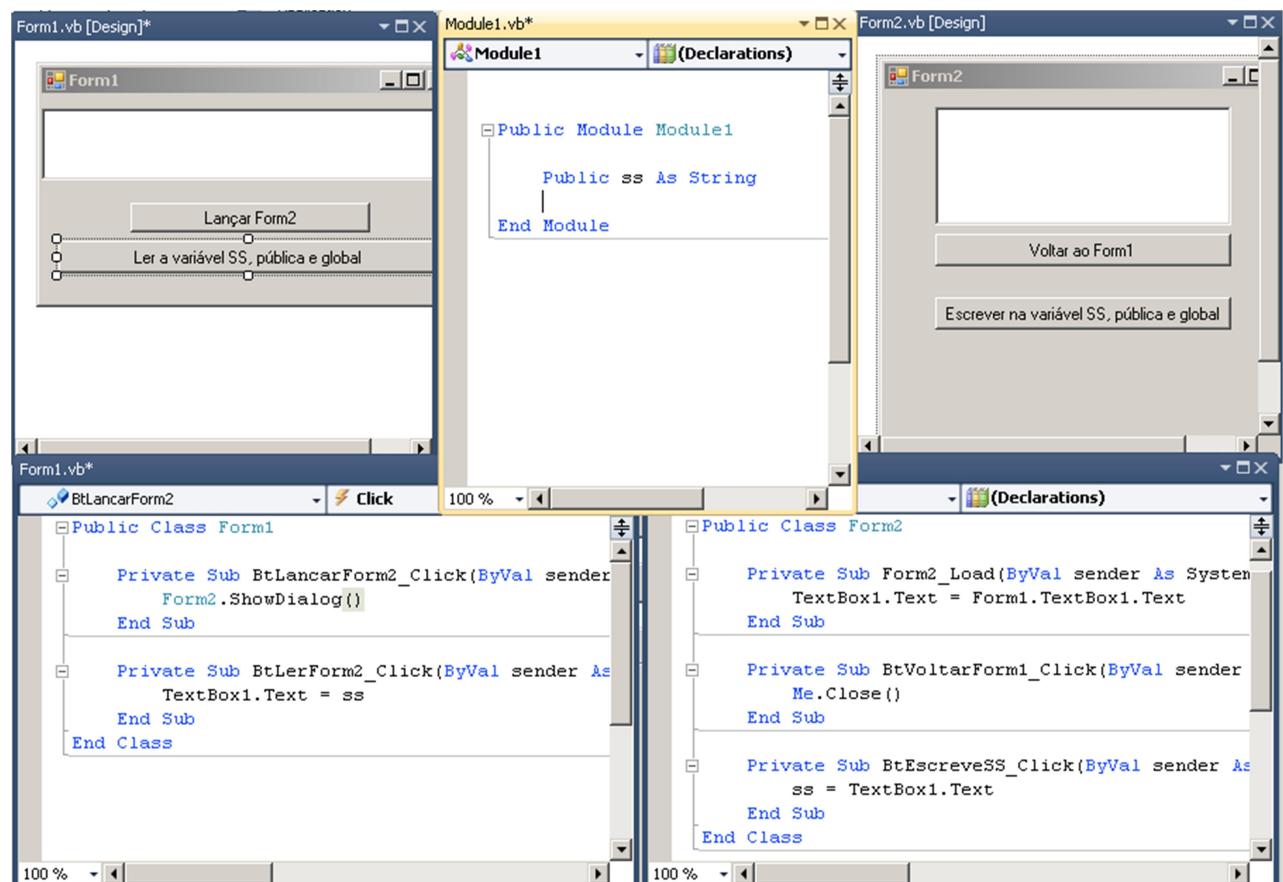


## 2.5.4 Módulos com código e dados

Pode também adicionar ao seu projeto um “**module**”. Ao contrário dos Forms, os módulos apenas permitem definir código e não têm uma interface gráfica associada.

O exemplo seguinte apresenta dois forms, o Form2 tem um botão que ao ser premido grava na variável global “SS” o texto que estiver na Form2.TextBox1.

O Form1 tem outro botão que lê a variável global e visualiza o texto na Form1.TextBox1.

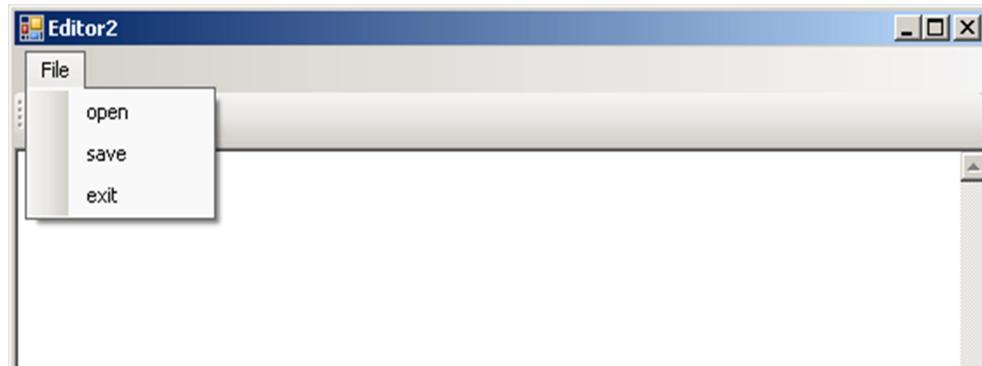




# Informática Industrial

## Trabalho prático 1 Editor de Texto

Pretende-se desenvolver um pequeno editor de texto que permita abrir e gravar ficheiros utilizando os objetos do tipo *MenuStrip*, *StatusStrip*, *OpenFileDialog*, *SaveFileDialog*, *StreamReader* e *StreamWriter*.



```
Imports System.IO
Public Class Editor2
    Dim le As StreamReader
    Dim escreve As StreamWriter

    Private Sub OpenToolStripMenuItem_Click(ByVal sender As System.Object,
        Dim s As String
        s = OpenFileDialog1.ShowDialog()

        If s = 1 Then
            le = New StreamReader(OpenFileDialog1.FileName)
            TextBox1.Text = le.ReadToEnd
            le.Close()
        End If

    End Sub

    Private Sub SaveToolStripMenuItem_Click(ByVal sender As System.Object,
        Dim s As String

        s = SaveFileDialog1.ShowDialog()
        If s = 1 Then
            escreve = New StreamWriter(SaveFileDialog1.FileName)
            escreve.WriteLine(TextBox1.Text)
            escreve.Close()
        End If

    End Sub

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object,
        End
    End Sub
```