

Laboratório de Programação I

Introdução à Programação em
Visual Basic

Fabricio Breve

Objetivos

- Ser capaz de escrever programas simples em Visual Basic
- Ser capaz de usar os comandos de entrada e saída
- Familiarizar-se com os tipos de dados
- Entender os conceitos básicos de memória
- Ser capaz de usar operadores aritméticos
- Entender a precedência dos operadores aritméticos
- Ser capaz de escrever comandos de tomada de decisão
- Ser capaz de usar os operadores de igualdade e os relacionamentos
- Ser capaz de usar os diálogos para exibir mensagens

Aplicativos de Console

- Contêm apenas saída de texto
- É um dos tipos de projeto mais simples
- A saída é exibida no *Prompt do MS-DOS* (Windows 95/98/ME) ou *Prompt de Comando* (Windows NT/2000/XP)

Programa Simples: Imprimindo uma Linha de Texto

```
1  ' Bemvindo.vb
2  ' Um programa simples em Visual Basic.
3
4  Module modFirstWelcome
5
6      Sub Main()
7          Console.WriteLine("Bem-Vindo ao Visual Basic!")
8      End Sub ' Main
9
10 End Module ' modFirstWelcome
```

Comentários

- Linhas 1 e 2 são linhas de comentários pois começam com um caractere de aspa simples (')
 - **Comentário de linha inteira:** aspa no início da linha
 - **Comentário de final de linha:** aspa no final de uma linha de código
- Linhas de comentários servem para melhorar a legibilidade do código
- Comentários são ignorados pelo compilador
- **Dica:** sempre inicie seu programa com um comentário descrevendo o mesmo

Definição de Módulo

- Linhas 4 e 10 definem nosso primeiro módulo
- Módulos são *agrupamentos lógicos de procedimentos* que simplificam a organização do programa
- Todo aplicativo console em VB tem no mínimo um módulo e um procedimento

Definição de Módulo

- **Module** é uma palavra-chave do VB
- O nome do nosso módulo (*modFirstWelcome*) é um identificador
- Identificadores:
 - Série de caracteres consistindo de letras, dígitos e caracteres de *underline* (_)
 - Não podem começar com dígitos nem conter espaços
- **Dica:** comece todo identificador de módulo com *mod* para facilitar sua identificação
- O VB não diferencia maiúsculas e minúsculas

Linhas em branco

- As linhas 3 e 5 são linhas em branco, ignoradas pelo compilador
- Servem apenas para melhorar a legibilidade do código

Main()

- A linha 6 está presente em todo aplicativo de console do VB
- É onde a execução do programa começa (ponto de entrada)
- Os parênteses no final de Main indicam que ele é um procedimento

Recuo

- Note que as linhas 6 a 8 estão recuadas em relação as outras
- Esta é uma convenção usada para melhorar a legibilidade do programa

Console.WriteLine

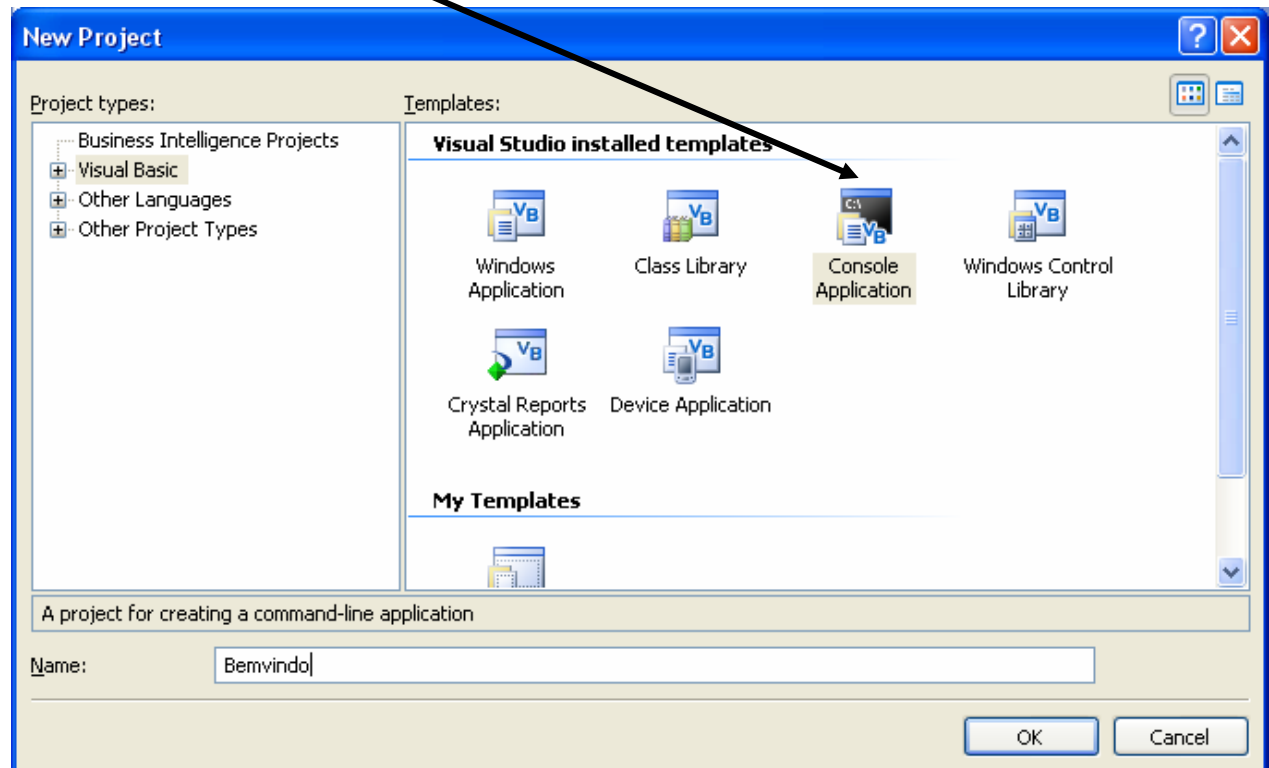
- A linha 7 é a que faz o trabalho real do programa (imprimir a série de caracteres entre aspas – *string*)
- A linha inteira é chamada *comando*
- Quando o comando é executado ele exibe a mensagem “Bem-Vindo ao Visual Basic”

Console.WriteLine

- Console.WriteLine são dois identificadores separados por um ponto
 - Console é o nome de uma classe e WriteLine é o nome de um método da classe Console
 - O método WriteLine imprime os caracteres passados como argumento e coloca o cursor na linha seguinte
- O programa termina quando encontra a linha 8 (*End Sub*)

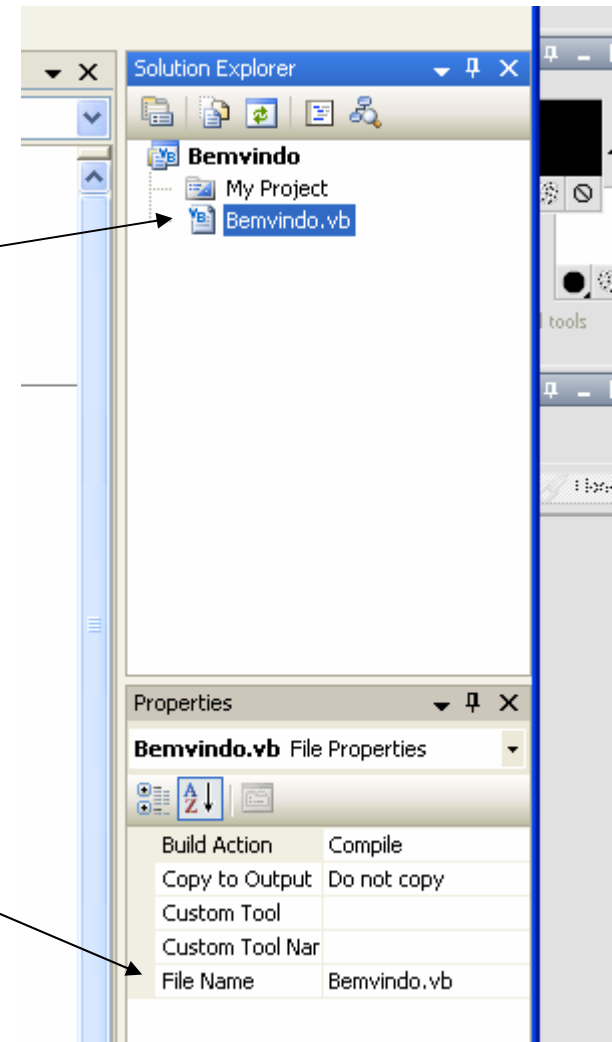
Implementando o programa

- Crie o aplicativo console:
 - File > New > Project
 - Console Application



Altere o nome do arquivo de programa

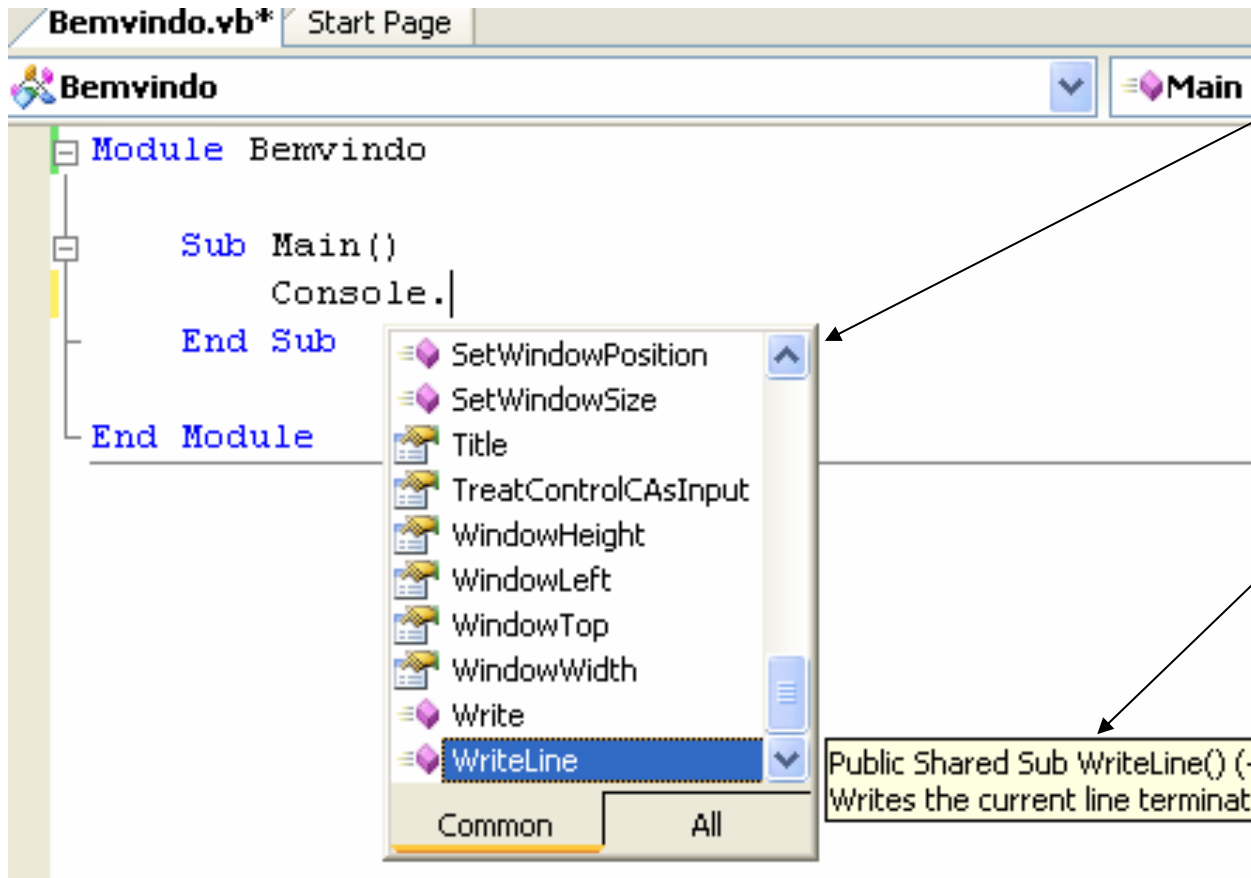
- Clique em *Module1.vb* no *Solution Explorer*
- Troque o nome na propriedade *File Name*



Escreva o código

- Escreva o código da linha 7

Lista de Membros



The screenshot shows the Visual Studio IDE with a code editor window titled 'Bemvindo.vb*' and a 'Start Page' tab. The code editor displays the following code:

```
Module Bemvindo  
    Sub Main()  
        Console.  
    End Sub  
End Module
```

A dropdown menu is open over the code editor, listing various members. The 'WriteLine' member is highlighted. The dropdown menu includes the following items:

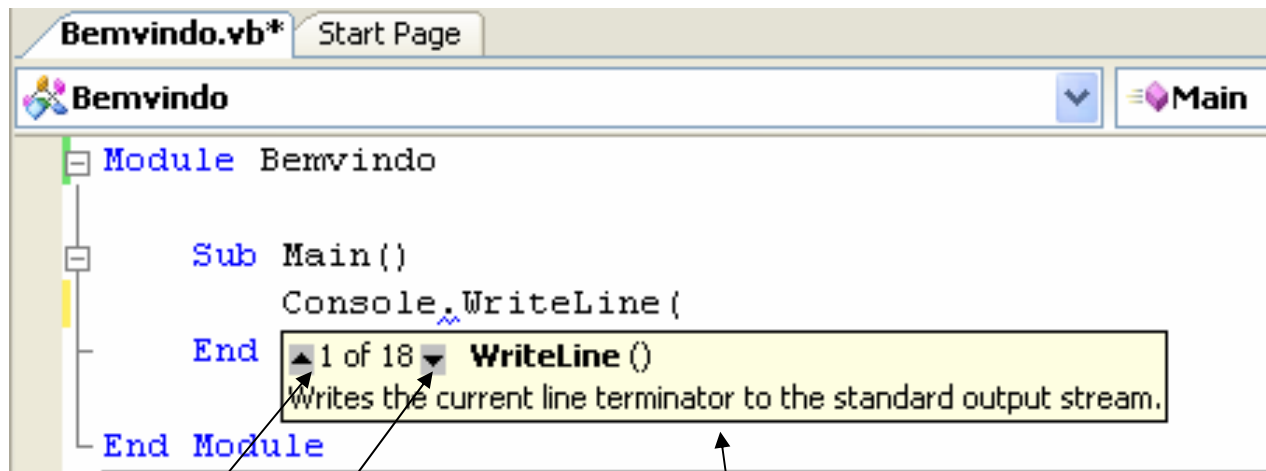
- SetWindowPosition
- SetWindowSize
- Title
- TreatControlCAsInput
- WindowHeight
- WindowLeft
- WindowTop
- WindowWidth
- Write
- WriteLine

The 'WriteLine' member is highlighted, and its description is shown in a tooltip:

```
Public Shared Sub WriteLine() (+  
Writes the current line terminatc
```

Descrição do membro
realçado

Escreva o código



The screenshot shows a Visual Studio code editor window titled "Bemvindo.vb*" with a "Start Page" tab. The code editor displays the following code:

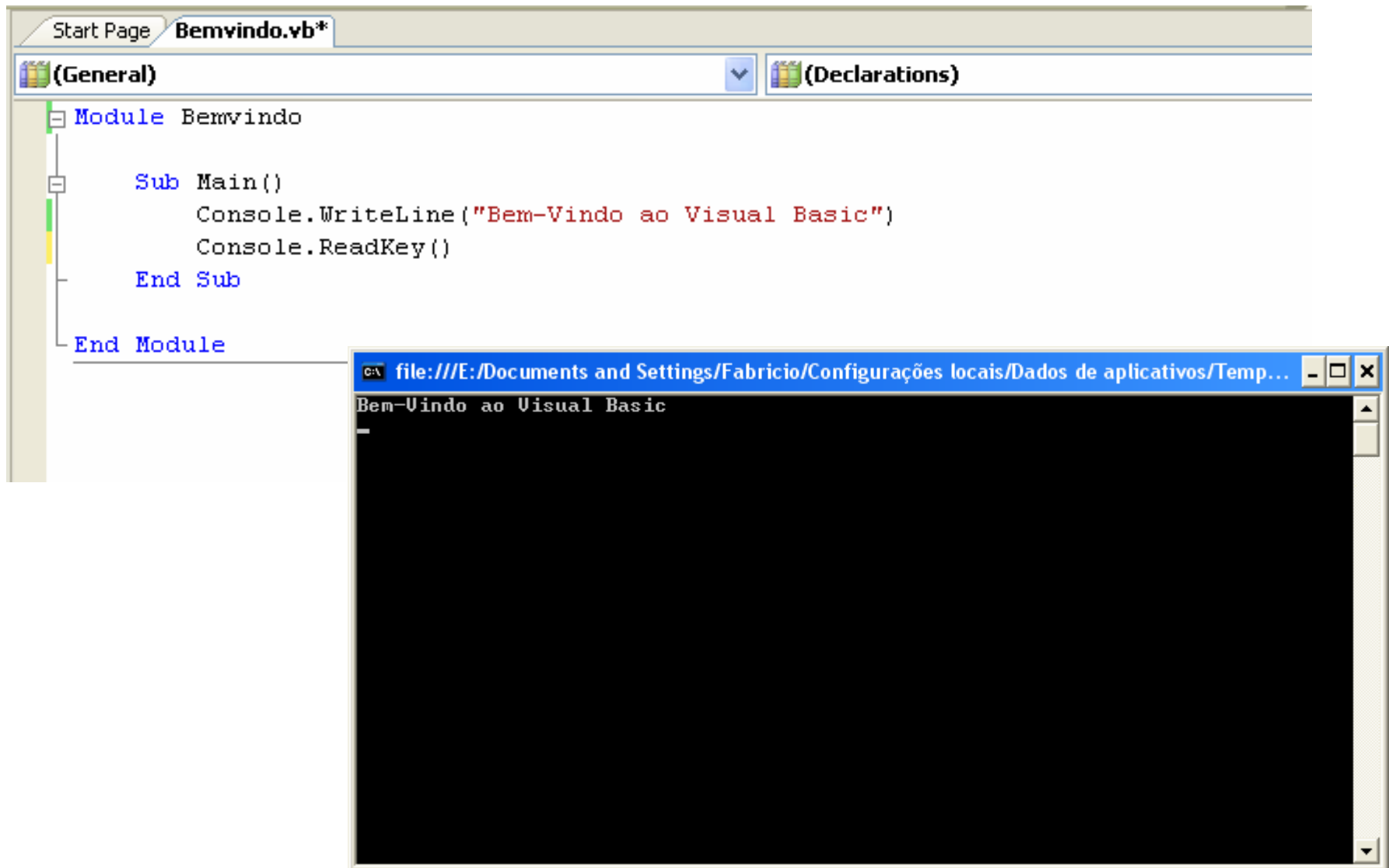
```
Module Bemvindo  
    Sub Main()  
        Console.WriteLine()  
    End Sub  
End Module
```

A tooltip is visible over the `WriteLine()` call, showing "1 of 18 WriteLine ()" and the description "Writes the current line terminator to the standard output stream." The tooltip is highlighted in yellow.

Setas para cima e para baixo

Janela Parameter Info

Compile e execute o Programa



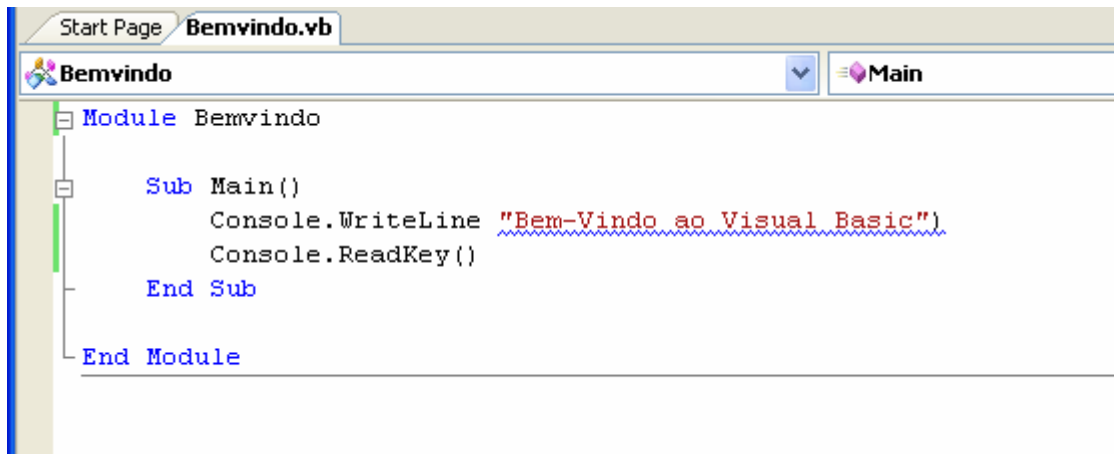
The image shows a screenshot of the Visual Studio IDE. The top window displays the code for a module named 'Bemvindo'. The code is as follows:

```
Module Bemvindo  
    Sub Main()  
        Console.WriteLine("Bem-Vindo ao Visual Basic")  
        Console.ReadKey()  
    End Sub  
End Module
```

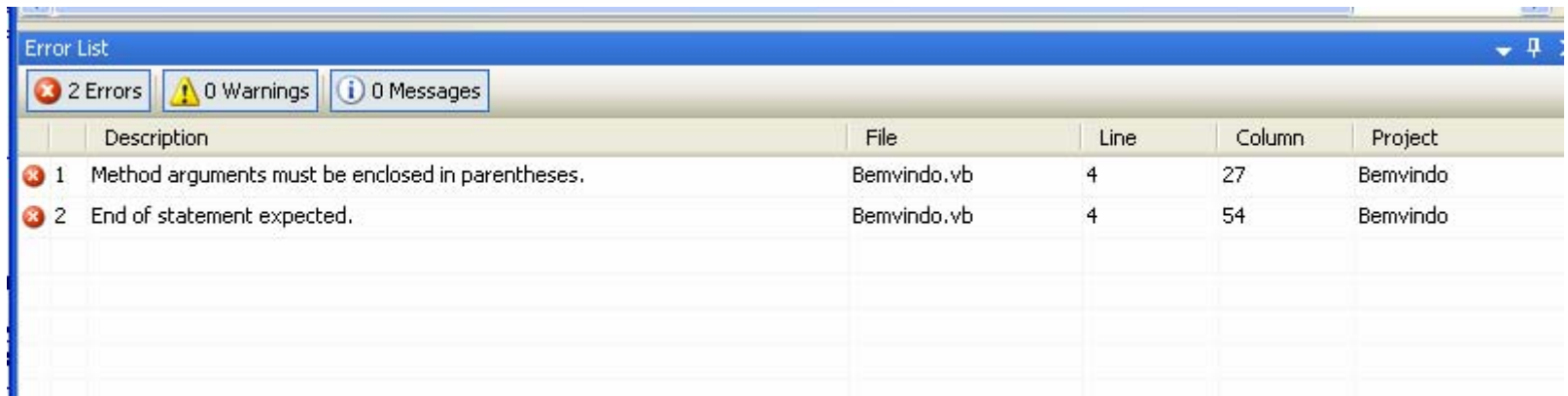
Below the code editor, a console window is open, showing the output of the program: "Bem-Vindo ao Visual Basic". The console window title is "file:///E:/Documents and Settings/Fabricio/Configurações locais/Dados de aplicativos/Temp...".

Erro de Sintaxe

- Aparece sublinhado em azul



```
Start Page Bemvindo.vb
Bemvindo
  Module Bemvindo
    Sub Main()
      Console.WriteLine "Bem-Vindo ao Visual Basic"
      Console.ReadKey()
    End Sub
  End Module
```



	Description	File	Line	Column	Project
✘ 1	Method arguments must be enclosed in parentheses.	Bemvindo.vb	4	27	Bemvindo
✘ 2	End of statement expected.	Bemvindo.vb	4	54	Bemvindo

Usando múltiplos comandos para imprimir uma linha de texto

```
' Escrevendo linha de texto com múltiplos comandos

Module BemVindo2

    Sub Main()
        Console.Write("Bem-Vindo ao ")
        Console.WriteLine("Visual Basic")
        Console.ReadKey()
    End Sub

End Module
```

Adicionando Inteiros

```
' Fig. 3.10: Adicao.vb
' Programa de Adição.

Module modAddition

    Sub Main()

        ' variáveis para guardar entrada do usuário
        Dim primeiroNumero, segundoNumero As String

        ' variáveis usadas no cálculo da adição
        Dim numero1, numero2, somaDosNumeros As Integer

        ' entrada do primeiro número pelo usuário
        Console.WriteLine("Por favor digite o primeiro inteiro: ")
        primeiroNumero = Console.ReadLine()

        ' ler segundo número digitado pelo usuário
        Console.WriteLine("Por favor digite o segundo inteiro: ")
        segundoNumero = Console.ReadLine()

        ' converter entradas para valores inteiros
        numero1 = primeiroNumero
        numero2 = segundoNumero

        somaDosNumeros = numero1 + numero2 ' adicionar números

        ' mostrar resultados
        Console.WriteLine("A soma é {0}", somaDosNumeros)
        Console.ReadKey()

    End Sub ' Main

End Module ' modAddition
```

Declarando variáveis

- As declarações começam com a palavra-chave *Dim*
- Os identificadores *primeiroNumero*, *segundoNumero*, *numero1*, *numero2* e *somaDosNumero* são variáveis (posições de memória do computador usadas pelo programa)
- Toda variável deve ser declarada antes de ser usada

Variáveis

- As variáveis *primeiroNumero* e *segundoNumero* são do tipo *String* (armazenam strings de caracteres)
- As variáveis *numero1* e *numero2* são do tipo *Integer*, ou seja, armazenam valores inteiros.
- O nome de uma variável pode ser qualquer identificador válido

Dicas

- Dê nome significativos para suas variáveis
- Por convenção nomes de variáveis são iniciados com letra minúscula, e cada palavra depois da primeira começa com letra maiúscula (Ex.: primeiroNumero)

ReadLine() e adição

- *numero1 = Console.ReadLine()*
 - método que interrompe o programa e aguarda uma entrada do usuário e atribui essa entrada à variável *numero1*
- *somaDosNumeros = numero1 + numero2*
 - Essa é a linha que pega os valores das variáveis *numero1* e *numero2*, os soma e atribui o resultado a variável *somaDosNumeros*

Imprimindo a soma

- *Console.WriteLine("A soma é {0}", somaDosNumeros)*
 - {0} será substituído pela variável somaDosNumeros
 - Outro exemplo:
 - *Console.WriteLine("Os números são {0}, {1} e {2}", numero1, numero2, numero3)*
 - {0}, {1} e {2} serão substituídos pelos valores das variáveis numero1, numero2 e numero3 respectivamente

Conceitos de Memória

- Variáveis
 - Correspondem a localizações reais na memória do computador
 - Toda variável tem
 - Nome
 - Tipo
 - Tamanho
 - Valor
 - Um valor atribuído a uma posição de memória substitui o valor que havia sido atribuído anteriormente
 - o valor anterior é destruído
 - Quando um valor é lido da memória ele não é destruído

Aritmética

- Operadores Aritméticos
 - O Visual Basic usa varios símbolos especiais não usados na álgebra
 - Asterisco (*), palavra-chave `Mod`
 - Operadores Binários
 - Operam usando dois operandos
 - soma + valor
 - Operadores Unários
 - Operadores que só usam um operando
 - +9, -19

Aritmética

- Divisão com resultado inteiro
 - Use a barra invertida, `\`
 - `7 \ 4` resulta em `1`
- Divisão com resultado em ponto-flutuante
 - Use a barra normal, `/`
 - `7 / 4` resulta em `1.75`
- Operador de Módulo, `Mod`
 - Mostra o resto de uma divisão inteira
 - `7 Mod 4` resulta em `3`

Operadores Aritméticos

Operação	Operador Aritmético	Expressão Algébrica	Expressão no Visual Basic
Adição	+	$f + 7$	$f + 7$
Subtração	-	$p - c$	$p - c$
Multiplicação	*	bm	$b * m$
Divisão (float)	/	X / y	x / y
Division (Integer)	\	Não tem	$v \setminus u$
Módulo	%	$r \text{ mod } s$	$r \text{ Mod } s$
Exponenciação	^	q^p	$q \wedge p$
Unário negativo	-	-e	-e
Unário Positivo	+	+g	+g

Regras de Precedência

1. Operadores em expressões contidas entre parênteses
2. Exponenciação
3. Unário positivo e negativo
4. Multiplicação e Divisão Ponto Flutuante
5. Divisão Inteira
6. Operação de módulo
7. Operações de adição e subtração

Precedência de Operadores

Operadores	Operação	Ordem de evolução (precedencia)
()	Parenteses	Avaliados primeiro. Se os parenteses estiverem aninhados, a expressão no par mais interno será avaliada primeiro. Se houverem vários pares de parenteses no mesmo nível (não aninhados), eles serão avaliados da esquerda para a direita.
^	Exponenciação	Avaliados em segundo lugar. Se houverem vários desses operadores, eles serão avaliados da esquerda para a direita.
+, -	Operadores de Sinal	Avaliados em terceiro lugar. Se houverem vários destes operadores, eles serão avaliados da esquerda para a direita.
*, /	Multiplicação e Divisão	Avaliados em quarto lugar. Se houverem vários destes operadores, eles serão avaliados da esquerda para a direita.
\	Divisão Inteira	Avaliados em quinto lugar. Se houverem vários destes operadores, eles serão avaliados da esquerda para a direita.
Mod	Módulo.	Avaliados em sexto lugar. Se houverem vários destes operadores, eles serão avaliados da esquerda para a direita.
+, -	Adição e Subtração	Últimos a serem avaliados. Se houverem vários destes operadores, eles serão avaliados da esquerda para a direita.

Exemplo de Precedência de Operadores

Passo 1. $y = 2 * 5 * 5 + 3 * 5 + 7$

$2 * 5$ é **10**

(multiplicação mais à esquerda)

Passo 2. $y = 10 * 5 + 3 * 5 + 7$

$10 * 5$ é **50**

(multiplicação mais à esquerda)

Passo 3. $y = 50 + 3 * 5 + 7$

$3 * 5$ é **15**

(Multiplicação antes da adição)

Passo 4. $y = 50 + 15 + 7$

$50 + 15$ é **65**

(Adição mais a esquerda)

Passo 5. $y = 65 + 7$

$65 + 7$ é **72**

(Última Adição)

Passo 6. $y = 72$ *(Última Operação – Atribuir 72 ao y)*

Estrutura de Decisão: If / Then

- **Estrutura If/Then**

- Permite que o programa tome uma decisão baseado na verdade ou falsidade de alguma expressão
- Condição
 - A expressão em uma estrutura **If/Then**
- Se a condição é verdadeira, os comandos que estiverem no corpo da estrutura serão executados
- Condições podem ser formadas usando
 - Operadores de igualdade
 - Operadores relacionais

Operadores de Igualdade e Relacionais

Operador de Igualdade ou Relacional Algébrico Padrão	Operadores de Igualdade ou Relacionais do Visual Basic	Exemplo de Condição do Visual Basic	Significado de Condição do Visual Basic
<i>Operadores de igualdade</i>			
=	=	x = y	x é igual a y
≠	<>	x <> y	x é diferente de y
<i>Operadores Relacionais</i>			
>	>	x > y	x é maior que y
<	<	x < y	x é menor que y
≥	>=	x >= y	x é maior ou igual a y
≤	<=	x <= y	x é menor ou igual a y

```
1 ' Fig. 3.19: Comparacao.vb
2 ' Usando operadores de igualdade e relacionais.
3
4 Module modComparison
5
6     Sub Main()
7
8         ' declare variáveis inteiras para entrada do usuário
9         Dim number1, number2 As Integer
10
11        ' leia o primeiro número digitado pelo usuário
12        Console.WriteLine("Por favor digite o primeiro inteiro: ")
13        number1 = Console.ReadLine()
14
15        ' leia o segundo número digitado pelo usuário
16        Console.WriteLine("Por favor digite o segundo inteiro : ")
17        number2 = Console.ReadLine()
18
19        If (number1 = number2) Then
20            Console.WriteLine("{0} = {1}", number1, number2)
21        End If
22
23        If (number1 <> number2) Then
24            Console.WriteLine("{0} <> {1}", number1, number2)
25        End If
26
27        If (number1 < number2) Then
28            Console.WriteLine("{0} < {1}", number1, number2)
29        End If
30
31        If (number1 > number2) Then
32            Console.WriteLine("{0} > {1}", number1, number2)
33        End If
```

Variáveis de mesmo tipo podem ser colocadas na mesma declaração

A estrutura If/Then compara se os valores de number1 e number2 são iguais

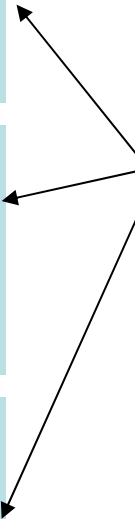
```
34
35 If (number1 <= number2) Then
36     Console.WriteLine("{0} <= {1}", number1, number2)
37 End If
38
39 If (number1 >= number2) Then
40     Console.WriteLine("{0} >= {1}", number1, number2)
41 End If
42
43 End Sub ' Main
44
45 End Module ' modComparison
```

```
Por favor digite o primeiro inteiro: 1000
Por favor digite o segundo inteiro: 2000
1000 <> 2000
1000 < 2000
1000 <= 2000
```

```
Por favor digite o primeiro inteiro:: 515
Por favor digite o segundo inteiro: 49
515 <> 49
515 > 49
515 >= 49
```

```
Por favor digite o primeiro inteiro:: 333
Por favor digite o segundo inteiro: 333
333 = 333
333 <= 333
333 >= 333
```

Saída do Programa



Precedência e Associatividade de Operadores

Operadores	Associatividade	Tipo
()	esquerda para a direita	parenteses
^	esquerda para a direita	exponenciação
* /	esquerda para a direita	multiplicativo e divisão ponto-flutuante
\	esquerda para a direita	divisão inteira
Mod	esquerda para a direita	módulo
+ -	esquerda para a direita	adição e subtração
= <> <	esquerda para a direita	igualdade e relacional
<= > >=		

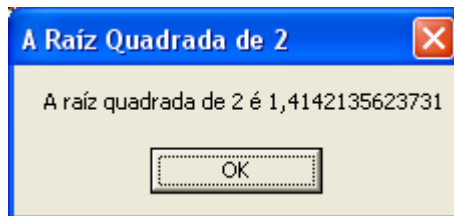
```
1 ' RaizQuadrada.vb
2 ' Mostra a raiz quadrada de 2 em um diálogo.
3
4 Imports System.Windows.Forms ' Namespace contendo MessageBox
5
6 Module modRaizQuadrada
7
8 Sub Main()
9
10 ' Calcula a raiz quadrada de 2
11 Dim root As Double = Math.Sqrt(2)
12
13 ' Mostra o resultado em um diálogo
14 MessageBox.Show("A raiz quadrada de 2 é " & root, _
15 "A Raiz Quadrada de 2")
16 End Sub ' Main
17
18 End Module ' modRaizQuadrada
```

método **Sqrt** da classe **Math** é chamado para calcular a raiz quadrada

Método **Show** da classe **MessageBox**

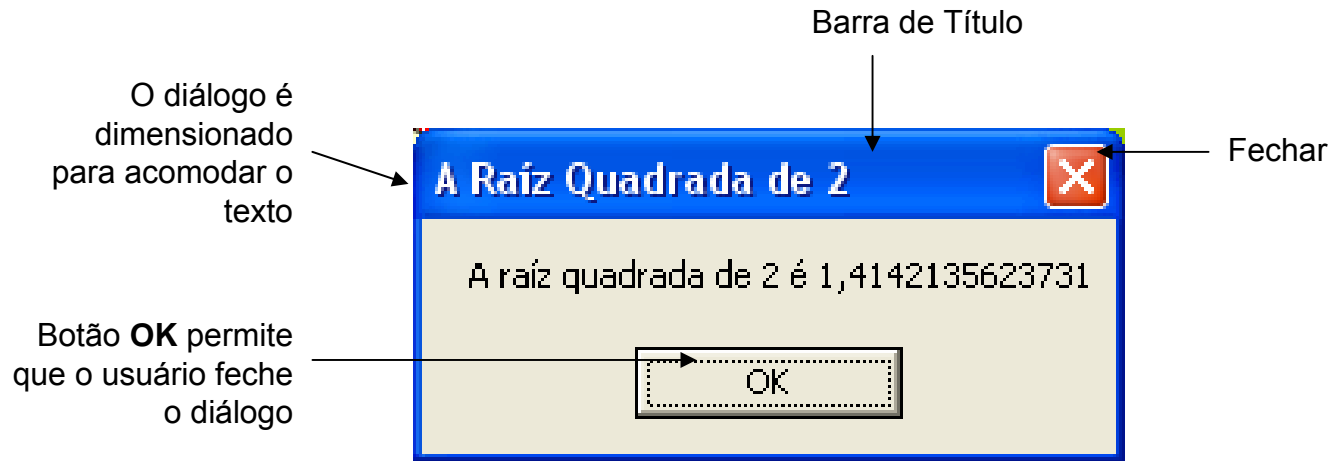
Caractere de continuação na próxima linha

Tipo **Double** armazena números de ponto flutuante



Saída do programa

Usando um diálogo para mostrar uma mensagem

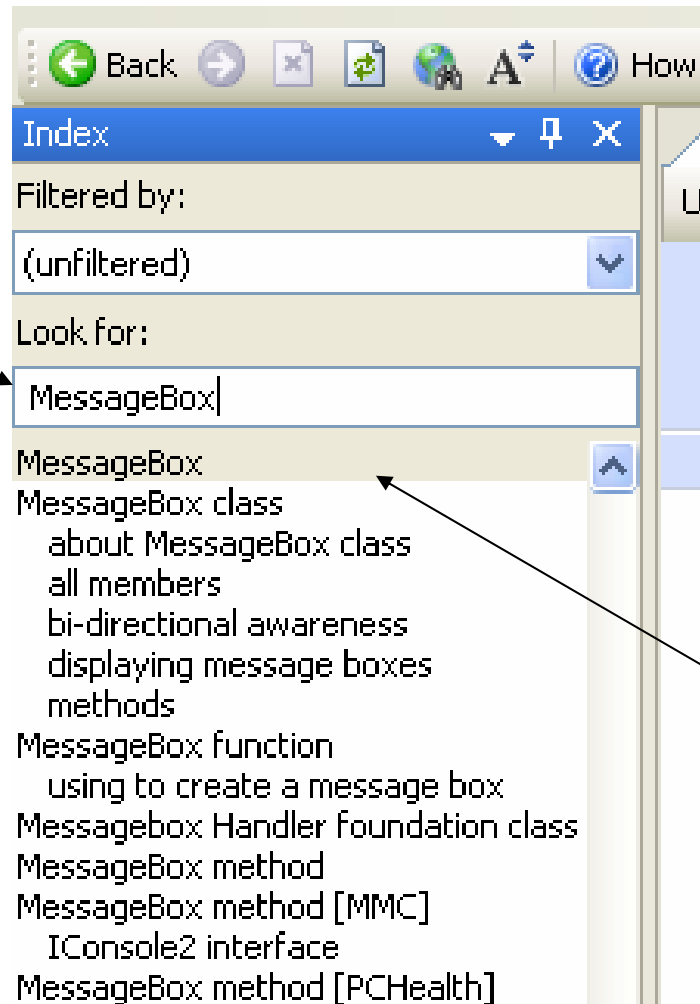


Usando um diálogo para mostrar uma mensagem

- Assembly
 - Arquivo que contém muitas classes fornecidas pelo Visual Basic
 - Estes arquivos tem a extensão `.dll` (dynamic link library)
 - Exemplo
 - A classe `MessageBox` está localizada no assembly `System.Windows.Forms.dll`
- Documentação MSDN
 - Informação sobre os assembly que nós precisamos podem ser encontradas na documentação no MSDN
 - Selecione **Help > Index...** para mostrar o diálogo **Index**

Acessando a documentação

String de Busca



Link para a documentação

Acessando a documentação

The screenshot shows the Visual Studio documentation for the `MessageBox` class. The left pane shows the index with the search term "MessageBox class, about MessageBox c". The main pane displays the class details, including the namespace and assembly information, which is circled in red.

MessageBox Class Search

URL: ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.NETDEVFX.v20.en/CPref17/html/T_System_Wind

.NET Framework Class Library

MessageBox Class

[See Also](#) [Example](#) [Members](#)

Collapse All Language Filter: All

Displays a message box that can contain text, buttons, and symbols that inform a

Namespace: System.Windows.Forms
Assembly: System.Windows.Forms (in system.windows.forms.dll)

Syntax

Visual Basic (Declaration)

```
Public Class MessageBox
```

Visual Basic (Usage)

```
Dim instance As MessageBox
```

C#

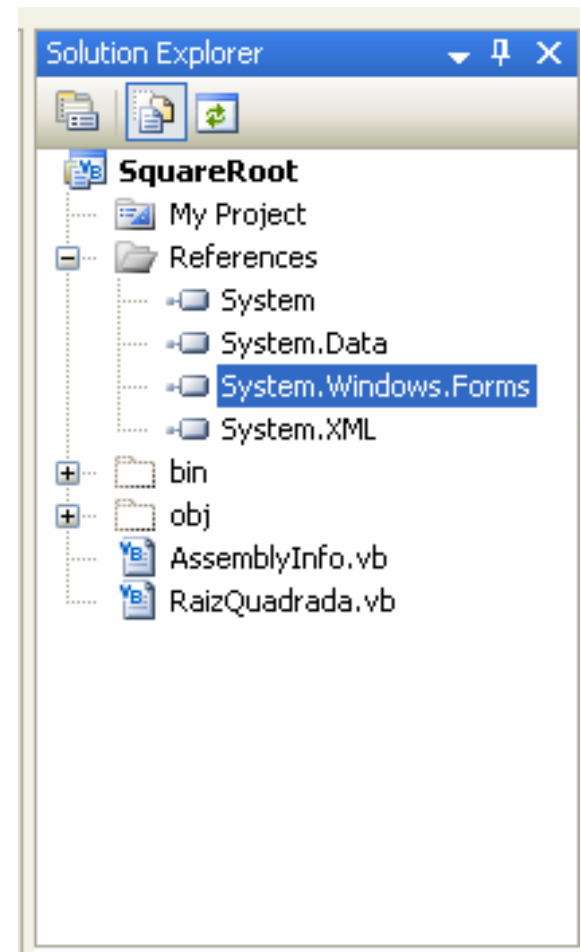
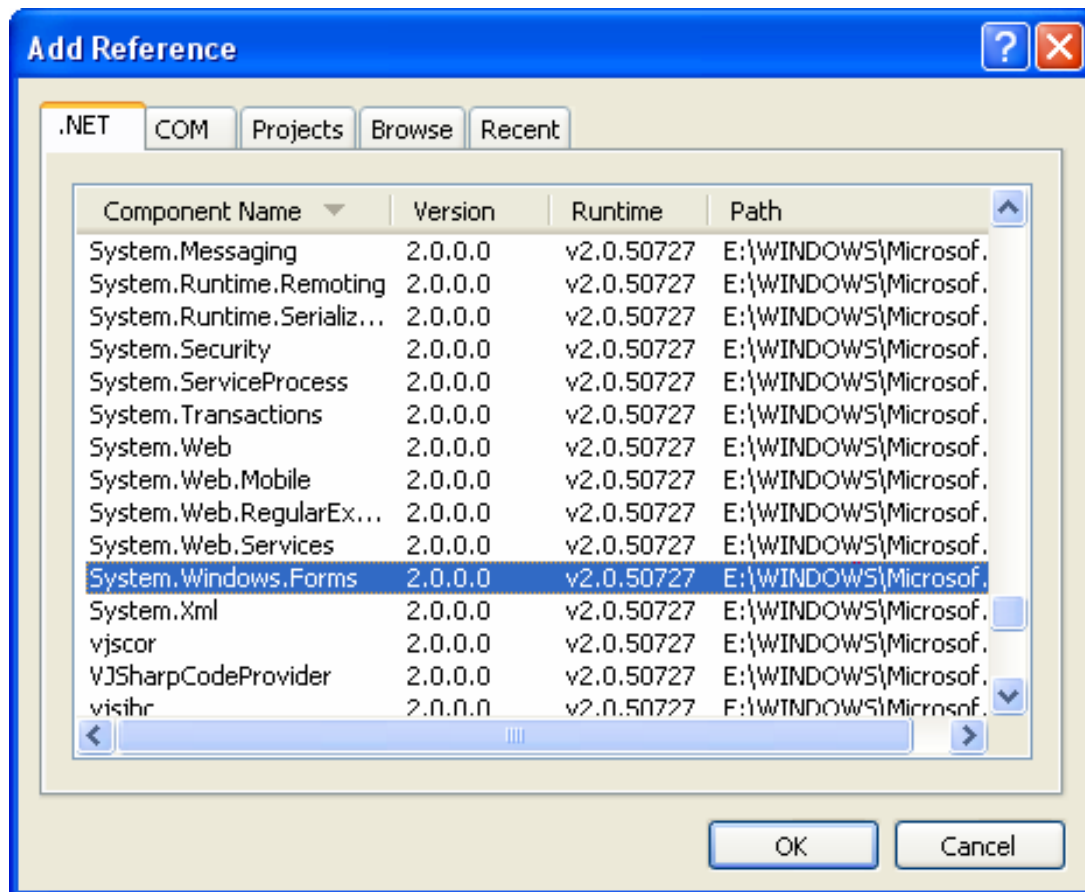
```
public class MessageBox
```

Referência / Imports

- Referência
 - É necessário adicionar uma referência ao assembly se você quiser usar suas classes
 - Exemplo
 - Para usar a classe **MessageBox** é necessário adicionar referência para **System.Windows.Forms**
- **Imports**
 - Esquecer de adicionar um comando **Imports** para um assembly referenciado gera um erro de sintaxe

Adicionando Referência

- **Project > Add Reference**



Exercícios

- 1) Exibir a mensagem “Olá” usando um MessageBox
- 2) O que é exibido no diálogo quando cada um dos seguintes comandos é executado?
Assuma que o valor de x é 2 e o valor de y é 3
 - a) `MessageBox.Show("x",x)`
 - b) `MessageBox.Show((x+x), "(x + x)")`
 - c) `MessageBox.Show("x+y")`
 - d) `MessageBox.Show(_
(x+y),(y+y))`

Exercícios

3) Qual o comando para a expressão

$$z = 8e^5 - n ?$$

4) Qual o valor de x para cada um dos comandos abaixo?

a) $X = 7 + 3 * 3 \setminus 2 - 1$

b) $X = 2 \text{ Mod } 2 + 2 * 2 - 2 / 2$

c) $X = (3 * 9 * (3 + (9 * 3 / (3))))$

Exercícios

- 5) Escreva um programa que solicite ao usuário que digite dois números e mostre a soma, a diferença, o produto e o quociente entre os dois números. Use a janela de comando para entrada e saída
- 6) Escreva um programa que receba como entrada do usuário o raio de um círculo e imprima o diâmetro, a circunferência e a área do círculo. Use as seguintes fórmulas (r é o raio):
diâmetro = $2r$, *circunferência* = $2\pi r$, *área* = πr^2 .
Use 3,14159 para π ou a constante Math.PI

Exercícios

7) Escreva um programa que leia dois inteiros, determine e imprima se o primeiro for um múltiplo do segundo. Por exemplo, se o usuário inserir 15 e 3, o primeiro número é um múltiplo do segundo. Se o usuário inserir 2 e 4, o primeiro não é um múltiplo do segundo. Use a janela de comando para entrada e saída. (Dica: use o operador de módulo)

Exercícios

8) Escreva um programa que receba como entrada do usuário um número consistindo de cinco dígitos, separe o número nos seus dígitos individuais e imprima os dígitos separados uns dos outros por três espaços em branco cada. Por exemplo, se o usuário digitar o número 42339, o programa deverá imprimir:

4 2 3 3 9

Referências Bibliográficas

- MSDN: <http://msdn.microsoft.com/vstudio/>
- [DEITEL, Harvey M.; DEITEL, Paul J.; NIETO, Tem R. *Véual Basic.NET: Como Programar*. Prentice-Hall, 2004](#)