



## **PRINCIPAIS NAMESPACES, ESTRUTURAS DE REPETIÇÃO**

**LUCAS CAMPOS – MCP.NET, MCAD**

**[CONTATO@LUCASCAMPOS.NET](mailto:CONTATO@LUCASCAMPOS.NET)**

**[LUCAS.CAMPOS@STUDENTPARTNERS  
.COM.BR](mailto:LUCAS.CAMPOS@STUDENTPARTNERS.COM.BR)**

# Agenda



- Objetivo do módulo;
- Principais Namespaces (Funções de Uso Geral);
- Exemplos destes Namespaces;
- Estrutura de Repetição - While;
- Exemplos – Estruturas de Repetição;

# Objetivo do módulo



- Apresentar aos participantes os principais namespaces existentes no .NET FRAMEWORK, bem como eles podem facilitar o desenvolvimento de aplicações, pois possuem várias classes e métodos que são de muita utilidade para nosso dia-a-dia.
- Apresentar a estrutura de repetição while.

# Principais Namespaces (Funções de Uso Geral)



- É interessante a partir desse ponto que seja apresentada ao usuário uma lista de algumas funções (métodos) de uso geral dentro da aplicação C#;
- O namespace System fornece, além das classes complexas, uma série de classes de uso geral que auxiliam enormemente o desenvolvimento de aplicações em modo geral;

# Principais Namespaces (Funções de Uso Geral)



- Namespace: `System.String`;
- Uma String é uma coleção sequencial de objetos da classe `System.Char` que vão gerar uma string;
- E uma string é uma coleção sequencial de caracteres, tipicamente utilizado para representar uma palavra, texto, etc...

# Principais Namespaces (Funções de Uso Geral)



- System.String – Principais métodos;
- ***Equals*** – Compara uma string com outra retornando true em caso de igualdade entre estas strings e false em caso de desigualdade. [Ex01](#).
- ***Substring*** – Copia a partir uma posição, um conjunto de caracteres da string. [Ex02](#).
- **Sintaxe:** x.Substring(posição inicial, posição final); [Ex03](#).

# Principais Namespaces (Funções de Uso Geral)



- ***Trim*** – Apaga os espaços em branco a direita e a esquerda da string.
- ***Replace*** – substitui um caracter ou uma seqüência por um outro caracter ou seqüência dentro da string. [Ex04](#).
- Namespace: System.Convert;
- Esta classe possui uma série de métodos específicos para a conversão de tipos;

# Principais Namespaces (Funções de Uso Geral)



- Estes métodos de uma forma ou de outra serão utilizados durante todo o decorrer deste curso;
- Esta classe retorna um valor que é equivalente ao valor específico de outro tipo;
- Os tipos suportados são: Boolean, Char, SByte, Byte, Int16, Int32, Int64, UInt16, UInt32, UInt64, Single, Double, Decimal, DateTime and String;

# Principais Namespaces (Funções de Uso Geral)



- *ToBoolean()* – converte para o tipo bool;
- *ToByte()* - converte para o tipo byte;
- *ToDateTime()* – converte para um tipo específico de data e hora ;
- *ToDecimal()* - converte para um valor real;

# Principais Namespaces (Funções de Uso Geral)



- *ToDouble()* – converte para o tipo double;
- *ToInt16()* – converte para inteiro em 16 bits;
- *ToInt32()* – converte para inteiro em 32 Bits;
- *ToInt64()* – converte para inteiro em 64 Bits;
- *ToSingle()* – converte para o tipo float;

# Principais Namespaces (Funções de Uso Geral)



- ToString() – converte para o tipo string;
- ToUInt16() – converte para inteiro em 16 bits positivo;
- ToUInt32() – converte para inteiro em 32 bits positivo;
- ToUInt64() – converte para inteiro em 64 bits positivo.

# Principais Namespaces (Funções de Uso Geral)



- Namespace: `System.Math`;
- Esta classe fornece uma grande gama de funções (métodos com retorno) ligado ao cálculo matemático.
- ***Abs()*** – retorna o módulo (valor absoluto de um valor).
- ***Max()*** – esta função retorna o maior valor entre dois valores.

# Principais Namespaces (Funções de Uso Geral)



- ***Min()*** – esta função retorna o menor valor entre dois valores;
- ***Pow()*** – esta função calcula a potência de **x** *elevado a y*;
- ***Sqrt()*** – retorna a raiz quadrada.
- [Ex05](#);

# Estrutura de Repetição



- Estruturas de repetições são instruções da linguagem, que permitem que um mesmo bloco de instruções execute continuamente por um conjunto de interações pré-determinadas;
- O C# possui uma gama bastante útil de loops (Como são chamadas as estruturas de repetições);

# Estrutura de Repetição - while



- A Palavra while significa “enquanto” e exprime a idéia de uma condição contínua.
- A sintaxe dessa estrutura é:
- ***while (condição)***
- ***{***
- ***... comando a serem executados caso a condição seja verdadeira....***
- ***}***
- [Ex06.](#)

# Estrutura de Repetição - while



- Análise da Estrutura While:
- Observe que o bloco de instruções será executado continuamente sempre que a condição analisada for verdadeira;
- Pela própria idéia do while, a partir do momento em que a condição analisada se tornar falsa o bloco deixa de ser executado;

# Treinamento.NET – namespaces e while



- N° 01: Somar a seqüência  $(1+2+3+4+5+6+\dots+99+100)$ .
- N° 03: Entre com uma palavra e retorne o seu inverso.
- Somar a seqüência  $(1-3+5-7+\dots+97-99)$ .

# Ex. 01 – System.String - Equals



```
• using System;
• class Ex01
• {
•     static void Main() // Entry point
•     {
•         string x = "maria";
•         string y = "joao";
•
•         if (x.Equals(y))
•             Console.WriteLine("São iguais");
•         else
•             Console.WriteLine("São diferentes");
•         Console.ReadLine();
•     }
• }
```

[Voltar](#)

# Ex. 02 – System.String - Substring



- using System;
- class Ex02
- {
- static void Main() // Entry point
- {
- string x = "maria josé";
- string y ;
- 
- y = x.Substring(6,4);
- Console.WriteLine("O valor copiado é {0}", y);
- Console.ReadLine();
- }
- } [Voltar](#)

## Ex. 03 – System.String - IndexOf



- using System;
- class Ex03
- {
- static void Main()
- {
- string x = "maria José";
- int y ;
- 
- y = x.IndexOf("José");
- Console.WriteLine("A Posição é {0}", y);
- Console.ReadLine();
- }
- }

[Voltar](#)

## Ex. 04 – System.String - Replace



- using System;
  - class Ex04
  - {
  - static void Main() // Entry point
  - {
  - string x = "maria José";
  - int y ;
  - 
  - y = x.Replace("José", "da Silva");
  - Console.WriteLine("O Novo nome é {0}", y);
  - Console.ReadLine();
  - }
  - }
- [Voltar](#)

# Ex. 05 – System.Math



```
• using System;
• class sistemat
• {
•     static void Main() // Entry point
•     {
•
•         double x, y;
•         x = Convert.ToDouble(Console.ReadLine());
•         y = Convert.ToDouble(Console.ReadLine());
•
•         Console.WriteLine("O módulo de {0} é {1}",x,Math.Abs(x));
•         Console.WriteLine("O maior é {0}", Math.Max(x,y));
•         Console.WriteLine("O menor é {0}", Math.Min(x,y));
•         Console.WriteLine(" {0} elevado a {1} é {2}",x,y,Math.Pow(x,y));
•         Console.WriteLine("A raiz quadrada de {0} é {1}",x,Math.Sqrt(x));
•         Console.ReadLine();
•     }
• }
```

[Voltar](#)

## Ex. 06 - while



- Entre com dez nomes.
  - using System;
  - class Ex01
  - {
  - static void Main() // Entry point
  - {
  - int contador = 1;
  - string nome;
  - while (contador <= 10)
  - {
  - nome = Console.ReadLine();
  - contador ++;
  - }
  - Console.ReadLine();
  - }
  - }
- [Voltar](#)

# Bibliografia



- Algoritmos Estruturados – 3<sup>a</sup> Edição Farrer Beckerm Faria – LTC;
- C#.NET – Guia do Desenvolvedor – Alta Books – 2<sup>a</sup> Edição;