# Linguagem de Programação 3

## Aula 3
## Funções Predefinidas
## da Biblioteca de Classes .NET

l.bertholdo@ifsp.edu.br

# Conteúdo

- Funções Predefinidas da Biblioteca de Classes .NET
  - Métodos para Funções Matemáticas
  - Métodos para Manipulação de Cadeias de Caracteres
  - Métodos para Manipulação de Data e Hora
  - Métodos para Formatação de Dados
  - Métodos para Conversão de Tipos

*l.bertholdo@ifsp.edu.br*

# Funções Predefinidas

- Classes e métodos nativos da biblioteca de classes .NET que visam simplificar e automatizar tarefas específicas comumente encontradas durante o desenvolvimento de aplicações.
- Por exemplo: arredondamento de números, cálculos matemáticos, formatação de dados, conversão de tipos de dados, manipulação de strings e datas.

# Métodos para Operações Matemáticas

- Métodos definidos na classe **System.Math** utilizados em expressões e operações matemáticas.
- Principais métodos:
  - Round
  - Sqrt
  - Pow

*l.bertholdo@ifsp.edu.br*

# Métodos para Operações Matemáticas

- Método Round
  - Arredonda números reais de acordo com uma quantidade especificada de casas decimais.

**Sintaxe:** System.Math.Round(<número>, [<casas decimais*>], [<modo de arredondamento para valores decimais que terminam com 5**>])

** Se não informado, retorna o número inteiro mais próximo.*
*** Se não informado, segue a mesma regra do parâmetro **MidpointRounding.ToEven**.*

```
Valor retornado: 1,37
Valor retornado: 1,9
Valor retornado: 1
Valor retornado: 1,4
Valor retornado: 1,5
Valor retornado: 1,4
Valor retornado: 1,4
```

```
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.372, 2));
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.88, 1));
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.3));

// AwayFromZero: Arredonda para o dígito mais afastado de zero.
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.35, 1, MidpointRounding.AwayFromZero));
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.45, 1, MidpointRounding.AwayFromZero));
// ToEven: Se o número à esquerda do “5” for par, arredonda para baixo. Se for ímpar, arredonda para cima.
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.35, 1, MidpointRounding.ToEven));
Console.WriteLine(" Valor retornado: " + System.Math.Round(1.45, 1, MidpointRounding.ToEven));
```

# Métodos para Operações Matemáticas

- Método Sqrt
  - Calcula a raiz quadrada de um número maior ou igual a zero.

    **Sintaxe:** System.Math.Sqrt(<radicando>)

```
Console.WriteLine(" Valor retornado: " + System.Math.Sqrt(81));
```

```
Valor retornado: 9
```

l.bertholdo@ifsp.edu.br

# Métodos para Operações Matemáticas

- Método Pow
  - Realiza operações de pontenciação.

    **Sintaxe:** System.Math.Pow(<base>, <expoente>)

```
Console.WriteLine(" Valor retornado: " + System.Math.Pow(5, 3));
```

```
Valor retornado: 125
```

  - Este método também pode ser usado para encontrar raízes com índice maior que 2, usando a sintaxe:

    **Sintaxe:** System.Math.Pow(<radicando>, 1.0/<índice>)

```
//Raíz cúbica de 27:
Console.WriteLine(" Valor retornado: " + System.Math.Pow(27, 1.0/3.0));

//Raíz quádrupla de 625:
Console.WriteLine(" Valor retornado: " + System.Math.Pow(625, 1.0/4.0));
```

```
Valor retornado: 3
Valor retornado: 5
```

l.bertholdo@ifsp.edu.br

# Métodos para Cadeias de Caracteres

- Métodos encapsulados pelo tipo de dados **string**. São utilizadas para manipular ou obter dados de cadeias de caracteres.
- Principais métodos:
  - Substring
  - Replace
  - Equals
  - Contains
  - StartsWith e EndsWith
  - IndexOf e LastIndexOf
- Propriedade Length

*l.bertholdo@ifsp.edu.br*

# Métodos para Cadeias de Caracteres

- Método Substring
  - Retorna parte de uma string com base em uma posição inicial e no número de caracteres que devem ser considerados a partir desta posição.

**Sintaxe:** <string>.Substring(<posição inicial>, [<número de caracteres*>])

** Se não informado, retorna até o último caractere presente na string.*

```
string texto = "R$150,00";
Console.WriteLine(texto.Substring(0, 5));
Console.WriteLine(texto.Substring(2, 3));
Console.WriteLine(texto.Substring(2));
```

```
R$150
150
150,00
```

l.bertholdo@ifsp.edu.br

# Métodos para Cadeias de Caracteres

- Método Replace
  - Substitui parte dos caracteres de uma string por outra.

**Sintaxe:** <string>.Replace(<string a ser substituída>, <string substituta>)

```
string texto = "Apartamento 53";
texto = texto.Replace("Apartamento", "Apto");
Console.WriteLine(texto);
```

```
Apto 53
```

l.bertholdo@ifsp.edu.br

# Métodos para Cadeias de Caracteres

- Método Equals
  - Compara variáveis ou valores do tipo **string** e retorna se os seus conteúdos são iguais ou não.
  - O método **Equals** é *case-sensitive*, ou seja, faz distinção entre letras maiúsculas e minúsculas.

```
string texto1 = "Livro";
string texto2 = "Livro";
string texto3 = "livro";
string texto4 = "Livro ";

Console.WriteLine(texto1.Equals(texto2));
Console.WriteLine(texto1.Equals(texto3));
Console.WriteLine(texto1.Equals(texto4));
```

```
True
False
False
```

l.bertholdo@ifsp.edu.br

# Métodos para Cadeias de Caracteres

- Método Contains
  - Retorna se uma cadeia de caracteres **contém** um caractere ou uma outra cadeia de caracteres.
  - O método **Contains** é *case-sensitive*, ou seja, faz distinção entre letras maiúsculas e minúsculas.

```
string texto = "Quando o Sol bater na janela do teu quarto";
Console.WriteLine(texto.Contains("jan"));
Console.WriteLine(texto.Contains("JAN"));
Console.WriteLine(texto.Contains("paz"));
```

```
True
False
False
```

l.bertholdo@ifsp.edu.br

# Métodos para Cadeias de Caracteres

- Métodos StartsWith e EndsWith
  - **StartsWith** retorna se uma cadeia de caracteres **inicia** com uma determinada string.
  - **EndsWith** retorna se uma cadeia de caracteres **termina** com uma determinada string.
  - Ambos métodos são *case-sensitive*. Para ignorar as distinções entre letras maiúsculas e minúsculas, é preciso usar o parâmetro **StringComparison.CurrentCultureIgnoreCase**.

```
string texto = "Quando o Sol bater na janela do teu quarto";
Console.WriteLine(texto.StartsWith("Q"));
Console.WriteLine(texto.StartsWith("q"));
Console.WriteLine(texto.StartsWith("q", StringComparison.CurrentCultureIgnoreCase));
Console.WriteLine(texto.StartsWith("X"));
Console.WriteLine(texto.EndsWith("to"));
Console.WriteLine(texto.EndsWith("TO"));
Console.WriteLine(texto.EndsWith("TO", StringComparison.CurrentCultureIgnoreCase));
Console.WriteLine(texto.EndsWith("z"));
```

```
True
False
True
False
True
False
True
False
```

# Métodos para Cadeias de Caracteres

- Métodos IndexOf e LastIndexOf
  - **IndexOf** retorna a posição da **1ª ocorrência** de uma string em uma cadeia de caracteres. E **LastIndexOf** retorna a posição da **última ocorrência** de uma string em uma cadeia de caracteres. Caso a string não seja encontrada, é retornado o valor **-1**.
  - Ambos métodos são *case-sensitive*. Para que eles procurem a string a partir de uma determinada posição, basta incluir um 2º argumento indicando esta posição.
  - Para ignorar as distinções entre letras maiúsculas e minúsculas, é preciso usar o parâmetro **StringComparison.CurrentCultureIgnoreCase**.

**Sintaxe:** <string>.IndexOf(<string a ser procurada>)
<string>.LastIndexOf(<string a ser procurada>)

```
string texto = "azul, verde, amarelo, azul, vermelho";
Console.WriteLine(texto.IndexOf("azul"));
Console.WriteLine(texto.IndexOf("azul", 2));
Console.WriteLine(texto.IndexOf("AZUL", StringComparison.CurrentCultureIgnoreCase));
Console.WriteLine(texto.LastIndexOf("azul"));
Console.WriteLine(texto.IndexOf("preto"));
```

```
0
22
0
22
-1
```

# Métodos para Cadeias de Caracteres

- Propriedade Length
  - Retorna o número de caracteres de uma string.

```
string texto = "Município de São Paulo";
Console.WriteLine(texto.Length);
```

```
22
```

l.bertholdo@ifsp.edu.br

# Métodos para Data e Hora

- Métodos e propriedades definidos na classe **DateTime** utilizados para manipular datas e horas.
- Principais métodos:
  - **AddYears** – Adiciona ou subtrai anos de uma data.
  - **AddMonths** – Adiciona ou subtrai meses de uma data.
  - **AddDays** – Adiciona ou subtrai dias de uma data.
  - **AddHours** – Adiciona ou subtrai horas de um horário.
  - **AddMinutes** – Adiciona ou subtrai minutos de um horário.
  - **AddSeconds** – Adiciona ou subtrai segundos de um horário.

*l.bertholdo@ifsp.edu.br*

# Métodos para Data e Hora

- Principais propriedades:
  - **Year** – Retorna o ano de uma data.
  - **Month** – Retorna o mês de uma data.
  - **Day** – Retorna o dia de uma data.
  - **Hour** – Retorna as horas de um horário.
  - **Minute** – Retorna os minutos de um horário.
  - **Second** – Retorna os segundos de um horário.
  - **DayOfWeek** – Retorna o dia da semana de uma data.

# Métodos para Data e Hora

```
DateTime data = new DateTime(2016, 2, 10, 15, 45, 22);
data = data.AddYears(2);
data = data.AddMonths(3);
data = data.AddDays(-4);
data = data.AddHours(-7);
data = data.AddMinutes(8);
data = data.AddSeconds(-22);

Console.WriteLine(data);
Console.WriteLine("Ano: " + data.Year);
Console.WriteLine("Mês: " + data.Month);
Console.WriteLine("Dia: " + data.Day);
Console.WriteLine("Horas: " + data.Hour);
Console.WriteLine("Minutos: " + data.Minute);
Console.WriteLine("Segundos: " + data.Second);
Console.WriteLine("Dia da semana: " + data.DayOfWeek);
```

```
06/05/2018 08:53:00
Ano: 2018
Mês: 5
Dia: 6
Horas: 8
Minutos: 53
Segundos: 0
Dia da semana: Sunday
```

l.bertholdo@ifsp.edu.br

# Métodos para Formatação

- Todos os tipos de dados possuem um método chamado **ToString**, que permite representá-los como uma string.
- Com isso, é possível formatar valores de diversas naturezas como números, valores monetários, percentuais, datas, etc.
- Além disso, também é possível formatar dados com o uso de **índices** e **máscaras**, que indicam a variável que deve ser formatada e seu respectivo formato.

# Métodos para Formatação

```
int numero1 = 23;
Console.WriteLine(numero1.ToString("000"));
Console.WriteLine("{0:000}", numero1); Console.WriteLine();

int numero2 = 48500;
Console.WriteLine(numero2.ToString("###,###,##0.00"));
Console.WriteLine("{0:###,###,##0.00}", numero2); Console.WriteLine();

Console.WriteLine("O cliente código {0:000} comprou\nR$ {1:###,###,##0.00} em mercadorias.",
                  numero1, numero2); Console.WriteLine();

float numero3 = 11.7f;
Console.WriteLine(numero3.ToString("0.00"));
Console.WriteLine("{0:0.00}", numero3); Console.WriteLine();

double porcentagem = 0.385;
Console.WriteLine(porcentagem.ToString("0.0%"));
Console.WriteLine("{0:0.0%}", porcentagem); Console.WriteLine();

long cpf = 12345678911;
Console.WriteLine(cpf.ToString(@"000\.000\.000-00"));
Console.WriteLine(@"{0:000\.000\.000-00}", cpf); Console.WriteLine();
```

máscara

índice

**0:** Indica que o dígito é obrigatório.
**#:** Indica que o dígito é opcional.

```
023
023

48.500,00
48.500,00

O cliente código 023 comprou
R$ 48.500,00 em mercadorias.

11,70
11,70

38,5%
38,5%

123.456.789-11
123.456.789-11
```

A **barra** indica que o ponto não é um separador de casa decimal e, por isso, não deve ser convertido em vírgula na impressão. O **"@"** indica que as barras devem ser desconsideradas na impressão.

*l.bertholdo@ifsp.edu.br*

# Métodos para Formatação

```
8/02/2016
08/02/2016
seg, 08/02/2016
segunda-feira, 08/02/2016

08/2/2016
08/02/2016
08/fev/2016
08/fevereiro/2016

08/02/16
08/02/2016
```

```
// Representações de dia
DateTime data = new DateTime(2016, 2, 8);
Console.WriteLine(data.ToString("d/MM/yyyy"));
Console.WriteLine(data.ToString("dd/MM/yyyy"));
Console.WriteLine(data.ToString("ddd, dd/MM/yyyy"));
Console.WriteLine(data.ToString("dddd, dd/MM/yyyy")); Console.WriteLine();

// Representações de mês
Console.WriteLine(data.ToString("dd/M/yyyy"));
Console.WriteLine(data.ToString("dd/MM/yyyy"));
Console.WriteLine(data.ToString("dd/MMM/yyyy"));
Console.WriteLine(data.ToString("dd/MMMM/yyyy")); Console.WriteLine();

// Representações de ano
Console.WriteLine(data.ToString("dd/MM/yy"));
Console.WriteLine(data.ToString("dd/MM/yyyy")); Console.WriteLine();
```

*l.bertholdo@ifsp.edu.br*

# Métodos para Conversão

- Métodos utilizados para conversão de tipos de dados.
- Por exemplo: string para números inteiros e reais, string para data, string para hora, float para inteiro, double para inteiro, char para string, números inteiros e reais para string, etc.

*l.bertholdo@ifsp.edu.br*

# Métodos para Conversão

```
string->int: 160
string->int: 160
string->long: 1234567890123456799
string->long: 1234567890123456799
string->float: 33,56
string->float: 33,56
string->double: 78369,17829
string->double: 78369,17829
string->decimal: 78369,17829
string->decimal: 78369,17829
string->DateTime: 10/02/2016 00:00:00
string->DateTime: 10/02/2016 00:00:00
string->DateTime: 8:25
string->DateTime: 8:25
```

*Os tipos **int**, **long**, **float**, **double**, **decimal** e **DateTime** são na verdade **structs**, estruturas que contém um conjunto de variáveis, cujos tipos de dados normalmente são diferentes. As structs são precursoras das **classes** existentes na programação orientada a objetos.*

```
// Conversão de string para int
string numero1 = "150";
Console.WriteLine("string->int: " + (int.Parse(numero1) + 10));
Console.WriteLine("string->int: " + (Convert.ToInt32(numero1) + 10));

// Conversão de string para long
string numero2 = "1234567890123456789";
Console.WriteLine("string->long: " + (long.Parse(numero2) + 10));
Console.WriteLine("string->long: " + (Convert.ToInt64(numero2) + 10));

// Conversão de string para float
string numero3 = "23,56";
Console.WriteLine("string->float: " + (float.Parse(numero3) + 10));
Console.WriteLine("string->float: " + (Convert.ToSingle(numero3) + 10));

// Conversão de string para double
string numero4 = "78359,17829";
Console.WriteLine("string->double: " + (double.Parse(numero4) + 10));
Console.WriteLine("string->double: " + (Convert.ToDouble(numero4) + 10));

// Conversão de string para decimal
string numero5 = "78359,17829";
Console.WriteLine("string->decimal: " + (decimal.Parse(numero5) + 10));
Console.WriteLine("string->decimal: " + (Convert.ToDecimal(numero5) + 10));

// Conversão de string para DateTime
string data = "10/02/2016";
Console.WriteLine("string->DateTime: " + DateTime.Parse(data));
Console.WriteLine("string->DateTime: " + Convert.ToDateTime(data));

// Conversão de string para DateTime
string hora = "08:25";
Console.WriteLine("string->DateTime: " +
    DateTime.Parse(hora).Hour + ":" + DateTime.Parse(hora).Minute);
Console.WriteLine("string->DateTime: " +
    Convert.ToDateTime(hora).Hour + ":" + Convert.ToDateTime(hora).Minute);
```

# Métodos para Conversão

```
int num1 = 36;
float num2 = 20.72f;
double num3 = 60.87;

// Conversão de float para int
int resultado1 = num1 + (int)num2;
Console.WriteLine("float -> int: " + resultado1);

// Conversão de double para int
int resultado2 = num1 + (int)num3;
Console.WriteLine("double -> int: " + resultado2);
```

```
float -> int: 56
double -> int: 96
```

```
// Conversão de char para string
char caractere = 'a';
string letra = caractere.ToString();
Console.WriteLine("char -> string: " + letra);

// Conversão de int para string
int n1 = 123;
string num = n1.ToString();
Console.WriteLine("int -> string: " + num);

// Conversão de float para string
float n2 = 157.33f;
num = n2.ToString();
Console.WriteLine("float -> string: " + num);

// Conversão de double para string
double n3 = 128.65;
num = n3.ToString();
Console.WriteLine("double -> string: " + num);
```

```
char -> string: a
int -> string: 123
float -> string: 157,33
double -> string: 128,65
```

*l.bertholdo@ifsp.edu.br*

# Referências


- Henrique Loureiro; C# 6.0 com Visual Studio – Curso Completo. FCA, 2015.
- John Sharp; Microsoft Visual C# 2013: Passo a Passo. Bookman, 2014.