LINGUAGEM DE PROGRAMAÇÃO II

Técnico em Informática Integrado ao Ensino Médio

Código: LP2 – Ano: 2º

Prof. Luiz Henrique Kiehn

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campos Cubatão, SP

CONTEÚDO DESTA AULA:

- Sobrecarga e Sobrescrita.
- Sobrescrevendo o método ToString().
- Formatação de saída:
  - Formatação de saída - ToString("...")
  - Personalização com "#" e "0"
  - Formatação por composição - {0,a:p}
  - Formatação por Interpolação - $"{var}"
  - String.Format
  - System.Globalization e CultureInfo

SOBRECARGA

- A **sobrecarga** de métodos permite ao programador definir diversos métodos **com o mesmo nome**, cada um apresentando elementos próprios.

SOBRECARGA

- Este recurso é usado quando se tem finalidades semelhantes, mas com elementos distintos.

- Um benefício é evitar que o programador busque nomes diferentes para finalidades semelhantes, o que, além de perda de tempo, poderia comprometer a legibilidade do código.

SOBRECARGA

- A sobrecarga de métodos pode-se dar quando houver **assinaturas** distintas para o mesmo nome de método. Uma assinatura pode diferir:
  1) pela quantidade de parâmetros passados para os métodos;
  2) pelas diferenças nos tipos de parâmetros passados, mesmo que a quantidade seja igual;
  3) pela ordem dos parâmetros, se é diferente entre os métodos considerados levando-se em conta os tipos.

Nota: Não são verificados tipos de retorno distintos. Nesse caso, será gerado erro na compilação.

SOBRECARGA – Assinatura difere pela quantidade de parâmetros

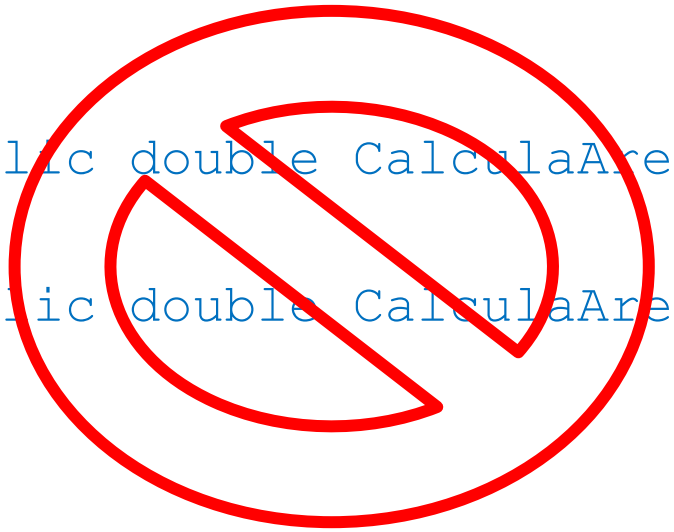
```
public double CalculaArea();

public double CalculaArea(string x);

public double CalculaArea(double x, double y);

public double CalculaArea(double x, double y, string z);
```

SOBRECARGA – Assinatura difere pelos tipos de parâmetros

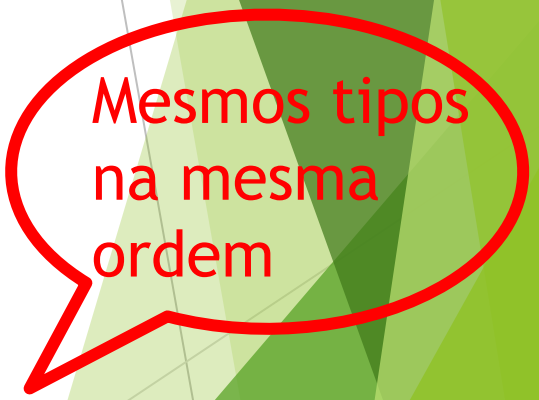
```
public double CalculaArea(string x);

public double CalculaArea(double x);

public double CalculaArea(int y);
```

SOBRECARGA – Assinatura difere pela ordem dos tipos de parâmetros

```
public double CalculaArea(double x, double y, string z);

public double CalculaArea(string z, double x, double y);
```



```
public double CalculaArea(double x, double y, string z);

public double CalculaArea(double a, double b, string c);
```



SOBRESCRITA

- **Sobrescrever** um método significa redefinir a ação e/ou resultado de um método previamente definido.

- Deve existir um método original com a mesma assinatura do método que irá sobrescrevê-lo.

SOBRESCRITA

- Para que isso seja possível, o método original deve ter sido definido com a cláusula **virtual** ou a cláusula **abstract**.

- Ao sobrescrever um método, a declaração deve conter a cláusula **override**.

SOBRESCRITA

```
// Método original
public virtual double CalculaArea() { ação... };

// Método que sobrescreve o original
public override double CalculaArea() { outra ação... };
```

SOBRESCREVENDO O MÉTODO ToString()

- Futuramente, voltaremos ao assunto sobrescrita quando falarmos de Polimorfismo.

- No momento, vamos ver a sobrescrita do método ToString() que, como já mencionado anteriormente, todo objeto possui, uma vez que toda classe herda a classe Object, que por sua vez possui o método ToString().

Fonte: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/how-to-override-the-tostring-method

SOBRESCREVENDO O MÉTODO ToString()

- O método ToString() permite retornar uma representação em formato **string** de qualquer objeto ou tipo.

```
int x = 10;
Console.WriteLine(x.ToString());
```

- Obs.: No exemplo abaixo, o método ToString() é chamado automaticamente.

```
Console.WriteLine(x);
```

SOBRESCREVENDO O MÉTODO ToString()

- No caso de objetos, não sendo o método ToString() sobrescrito, retorna-se a identificação do objeto.

```
Namespace Funcionario {...
Pessoa pessoa1 = new Pessoa();
pessoa1.Nome = "Mahatma Ghandi";
pessoa1.DataNascimento = DateTime.Parse("02/10/1869");
Console.WriteLine(pessoa1.ToString());

// Apresenta na tela:
// Funcionario.Pessoa
```

SOBRESCREVENDO O MÉTODO ToString()

- Considerando uma classe Pessoa com os atributos Nome e DataNascimento, podemos ter o método ToString() sobrescrito desta forma:

```
public override string ToString(){
   return "Nome: " + Nome + "\n "
         + "Data nascimento: "
            DataNascimento.ToString();
}
```

- Neste caso, no exemplo do *slide* anterior, teremos a seguinte saída:

Mahatma Ghandi 02/10/1869 00:00:00

FORMATAÇÃO DE SAÍDA

- Existem diversas maneiras de formatar uma saída (em tela, em impressão etc.).
- Vamos ver algumas aqui, mas ressaltando que há muitas possibilidades conforme o tipo de variável (Int, Double, DateTime, Temperature, etc.).
- Para cada necessidade, recomenda-se consultar a literatura correspondente, para aprofundar-se no assunto.
- Neste e nos próximos *slides* há *links* sobre a questão.

Fontes: https://learn.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
https://learn.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings

FORMATAÇÃO DE SAÍDA – Tipos Numéricos

- A tabela abaixo apresenta alguns parâmetros para formatação de tipos numéricos (inteiros, float, double, decimais etc.) usados no método ToString().

Parâmetro	Descrição	Exemplo de saída
“B” ou “b”	Binário	73 (“B”) -> 01001001
“C” ou “c”	Monetário (*Currency*)	123.45 (“C”) -> $123,45
“D” ou “d”	Decimal (apenas inteiros)	-1234 (“D6”) -> -001234
“E” ou “e”	Exponencial	2049.345 (“E”) -> 2,049345E+003
“F” ou “f”	Ponto decimal	-1234.56(“F4”) -> -1234,5600
“N” ou “n”	Ponto decimal com separador de milhar	-1234.56(“N4”) -> -1.234,5600
“P” ou “p”	Percentual	0,4268 (“P1”) -> 42,7%
“X” ou “x”	Hexadecimal	255 (“X”) -> FF

Exemplo: valor.ToString(“N3”) => 18.593

FORMATAÇÃO DE SAÍDA – Tipos Numéricos

- Também é possível efetuar uma formatação personalizada utilizando os símbols **"#"**, **"0"**, **","** e **"."**.

Símbolo	Descrição	Exemplo de saída
"0"	Apresenta o dígito ou zero na respectiva posição.	0.387 ("0.00") -> 0,39
"#"	Apresenta o dígito ou espaço na respectiva posição. Também usado para grupos de milhares.	0.387 ("#.##") -> ,39
"."	Determina a posição do ponto decimal.	0.387 ("0.0000") -> 0,3870
","	Separador de grupos de milhares.	123456789 ("##,#") -> 123.456,789

Fonte: https://learn.microsoft.com/en-us/dotnet/standard/base-types/custom-numeric-format-strings

FORMATAÇÃO DE SAÍDA
Formatação de Composição

- Outra forma de se formatar valores é através da Formatação de Composição.

```
Produto prod1 = new Produto("Arroz", 18.52);
Console.WriteLine("Produto: {0}, preço: {1:C}.",
                  prod1.Descricao,
                  prod1.Preco);
```

Saída:

Produto: Arroz, preço: R$18,52.

Fonte: https://learn.microsoft.com/pt-br/dotnet/standard/base-types/composite-formatting

FORMATAÇÃO DE SAÍDA
Formatação de Composição - Alinhamento

- Pode-se alinhar os campos à esquerda e à direita.

```
Produto prod1 = new Produto("Arroz", 18.52);
Console.WriteLine("{0,-20} {1,-8}.", "Produto:", "Preço: ");
Console.WriteLine("{0,-20} {1,8:C}", prod1.Descricao, prod1.Preco);
```

Saída:

```
Produto:              Preço:
Arroz                    R$18,52
```

Obs.:

Valor negativo: alinhamento à esquerda. Usado para strings, entre outros.
Valor positivo: alinhamento à direita. Usado para números, entre outros.

FORMATAÇÃO DE SAÍDA
Interpolação de Cadeias de Caracteres ($)

- Também pode-se formatar cadeia de caracteres por Interpolação de Cadeias de Caracteres usando o caractere “$”.

```
Produto prod1 = new Produto(“Arroz”, 18.52);
Console.WriteLine($“Produto: {prod1.Descricao,-20},
                  preço: {prod1.Preco,8:C}.“);
```

Saída:

Produto: Arroz preço: R$18,52.

Fonte: https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/tokens/interpolated

FORMATAÇÃO DE SAÍDA - String.Format

- O método StringFormat recebe como parâmetros a formatação desejada (utilizando a formatação por composição) e uma lista de valores.

```
double valor = 123.4567;
string str = String.Format("| {0,10:C1}", valor);
Console.WriteLine(str);
```

Fonte: https://learn.microsoft.com/pt-br/dotnet/api/system.string.format?view=net-8.0

FORMATAÇÃO DE SAÍDA
System.Globalization e CultureInfo

- Pode-se, também, especificar a formatação conforme as convenções de cada língua e/ou país.

```
decimal value = 1603.42m;
Console.WriteLine(value.ToString("C3", new CultureInfo(“pt-BR")));
Console.WriteLine(value.ToString("C3", new CultureInfo("en-US")));
Console.WriteLine(value.ToString("C3", new CultureInfo("fr-FR")));
Console.WriteLine(value.ToString("C3", new CultureInfo("de-DE"))); } }
// Saídas:
// R$1.603,420
// $1,603.420
// 1 603,420 €
// 1.603,420 €
```

Fonte: https://learn.microsoft.com/en-us/dotnet/standard/base-types/formatting-types#culture-sensitive-formatting-with-format-providers

FORMATAÇÃO DE SAÍDA
System.Globalization e CultureInfo

- Para se usar a formatação pelo método CultureInfo, é necessário importar a classe System.Globalization.

```
using System.Globalization;
```

- Nos formatos anteriores, também se pode usar o recurso CultureInfo. Segue exemplo:

```
integerNumber = -29541;
Console.WriteLine(integerNumber.ToString("N3", CultureInfo.InvariantCulture));
// Saída:
    -29,541.000
```

Obs.: Invariant.Culture representa o formato padrão dos EUA.

FORMATAÇÃO DE SAÍDA
System.Globalization e CultureInfo

- Outro exemplo:

```
CultureInfo ci = new CultureInfo("en-us");
double valor = 10761.937554;
Console.WriteLine("C: {0}", valor.ToString("C", ci));

// Displays "C: $10,761.94"
```

TRABALHO 1º Bimestre – parte 1/3
TABELA DE PRODUTOS

- Elabore um programa que gere como saída a tabela abaixo.

Código	Descrição	Preço	Quant.Estoque
7891025101604	Leite	R$ 3.00	15.00 1
7891000105016	Barra cereal	R$ 3.40	28.00 lt
7891000120101	Creme de leite	R$ 3.70	23.00 cx
7891000100103	Leite condensado	R$ 4.50	18.00 und
7891999000538	Iogurte	R$ 1.98	32.00 gf
7896051126041	Leite fermentado	R$ 2.12	12.00 cx
7897236904805	Água	R$ 1.50	48.00 cp
7622300830083	Biscoito recheado	R$ 1.80	35.00 pct
7891150036567	Caldo de galinha	R$ 1.90	16.00 cx
4005900036728	Desodorante	R$ 11.10	25.00 und
7896185989819	Vitamina C	R$ 35.20	26.00 und
7898113811452	Lanche	R$ 9.50	37.00 und

TRABALHO 1º Bimestre – parte 1/3
TABELA DE PRODUTOS

- Os campos da tabela devem estar devidamente formatados e alinhados.
- Textos (código, descrição e unidade) devem estar alinhados à esquerda.
- Valores (quantidade e preço) devem estar alinhados à direita.
- Valores monetários devem ser precedidos pelo símbolo R$.
- **Os códigos e descrições devem ser os especificados conforme consta na imagem do *slide* anterior.**

TRABALHO 1º Bimestre – parte 1/3
TABELA DE PRODUTOS

// Tabela de Produtos - Código EAN13

01. "7891025101604"
02. "7891000105016"
03. "7891000120101"
04. "7891000100103"
05. "7891999000538"
06. "7896051126041"
07. "7897236904805"
08. "7622300830083"
09. "7891150036567"
10. "4005900036728"
11. "7896185989819"
12. "7898113811452"

// Tabela de Produtos - Descrição

01. "Leite"
02. "Barra cereal"
03. "Creme de leite"
04. "Leite condensado"
05. "Iogurte"
06. "Leite fermentado"
07. "Água"
08. "Biscoito recheado"
09. "Caldo de galinha"
10. "Desodorante"
11. "Vitamina C"
12. "Lanche"