

# Lógica e Pensamento Computacional II

## com C#

---

Prof. Waldeck Lindoso Jr.

# Tipos básicos (Valor)

C# Type	.Net Framework Type	Signed	Bytes	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	$-2^{31}$ to $2^{31} - 1$
long	System.Int64	Yes	8	$-2^{63}$ to $2^{63} - 1$
byte	System.Byte	No	1	0 to 255
ushort	System.UInt16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to $2^{32} - 1$
ulong	System.UInt64	No	8	0 to $2^{64} - 1$
float	System.Single	Yes	4	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ with 7 significant figures
double	System.Double	Yes	8	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character
bool	System.Boolean	N/A	1/2	true or false

# Tipos básicos (**Referência**)

Tipo C#	Tipo .NET	Descrição
string	System.String	Uma cadeia de caracteres Unicode <b>IMUTÁVEL</b> ( <i>segurança, simplicidade, thread safe</i> )
object	System.Object	Um objeto genérico (toda classe em C# é subclasse de object) GetType Equals GetHashCode ToString

# praticando...

```
bool completo = false;
char genero = 'F';
char letra = '\u0041'; * através do unicode table
byte n1 = 126;
int n2 = 1000;
int n3 = 2147483647;
long n4 = 2147483648L;
float n5 = 4.5f;
double n6 = 4.5;
String nome = "Maria Green";
Object obj1 = "Alex Brown";
Object obj2 = 4.5f;
```

\* **Object** é um tipo genérico e aceita qualquer tipo(string, float, ...).

```
Console.WriteLine(completo);
Console.WriteLine(genero);
Console.WriteLine(letra);
Console.WriteLine(n1);
Console.WriteLine(n2);
Console.WriteLine(n3);
Console.WriteLine(n4);
Console.WriteLine(n5);
Console.WriteLine(n6);
Console.WriteLine(nome);
Console.WriteLine(obj1);
Console.WriteLine(obj2);
```

Microsoft Visual Studio Debug Console

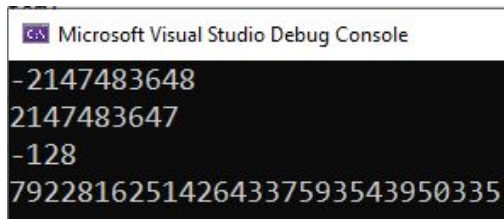
```
False
F
A
126
1000
2147483647
2147483648
4,5
4,5
Maria Green
Alex Brown
4,5
```

# Funções para números mínimos e máximos

- `int.MinValue`
- `int.MaxValue`
- `sbyte.MaxValue`
- `long.MaxValue`
- `decimal.MaxValue`
- etc...

```
int n1 = int.MinValue;  
int n2 = int.MaxValue;  
sbyte n3 = sbyte.MinValue;  
decimal n4 = decimal.MaxValue;
```

```
Console.WriteLine(n1);  
Console.WriteLine(n2);  
Console.WriteLine(n3);  
Console.WriteLine(n4);
```



Microsoft Visual Studio Debug Console

```
-2147483648  
2147483647  
-128  
79228162514264337593543950335
```

# Restrições para nomes de variáveis

- Não pode começar com dígito: use uma letra ou \_
- Não usar acentos ou til
- Não pode ter espaço em branco
- Sugestão: use nomes que tenham um significado

```
// Modo Correto
int _5minutos;
int salario;
int salarioDoFuncionario;

// Modo Errado
int 5minutos;
int salário;
int salario do funcionario;
```

# Convenções

- Camel Case: `lastName` (parâmetros de métodos, variáveis dentro de métodos)
- Pascal Case: `LastName` (namespaces, classe, properties e métodos)
- Padrão `_lastName` (atributos "internos" da classe)

# Exemplo de convenção

```
namespace Curso
{
    class ContaBancaria
    {
        public string Titular { get; set; }
        private double _saldo;

        public void Deposito(double quantia)
        {
            _saldo += quantia;
        }

        public double GetSaldo()
        {
            return _saldo;
        }
    }
}
```



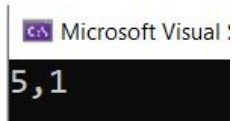
# Conversão implícita e casting

- Conversão implícita entre tipos
  - <https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/implicit-numericconversions-table>
- Casting: conversão explícita entre tipos COMPATÍVEIS

# Conversão implícita e casting

## Exemplo 1

```
double a;  
float b;  
a = 5.1;  
b = (float)a;  
Console.WriteLine(b);
```

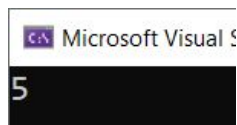


Microsoft Visual S

5,1

## Exemplo 2

```
double a;  
int b;  
a = 5.1;  
b = (int)a;  
Console.WriteLine(b);
```



Microsoft Visual S

5

# Conversão implícita e casting

## Exemplo 3

```
int a = 5;  
int b = 2;  
double resultado = (double)a / b;  
Console.WriteLine(resultado);
```



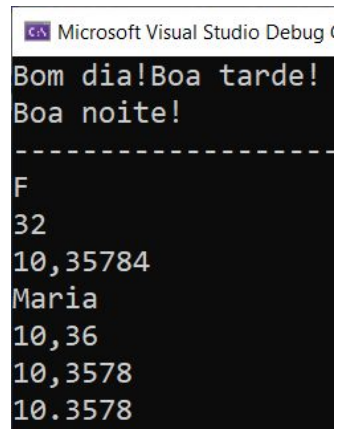
# Saída de dados (**console**)

- Comandos
  - `Console.WriteLine( valor );`
  - `Console.Write( valor );`

## praticando...

```
char genero = 'F';
int idade = 32;
double saldo = 10.35784;
String nome = "Maria";

Console.Write("Bom dia!");
Console.WriteLine("Boa tarde!");
Console.WriteLine("Boa noite!");
Console.WriteLine("-----");
Console.WriteLine(genero);
Console.WriteLine(idade);
Console.WriteLine(saldo);
Console.WriteLine(nome);
Console.WriteLine(saldo.ToString("F2"));
Console.WriteLine(saldo.ToString("F4"));
Console.WriteLine(saldo.ToString("F4", CultureInfo.InvariantCulture));
```



Microsoft Visual Studio Debug Console

```
Bom dia!Boa tarde!
Boa noite!
-----
F
32
10,35784
Maria
10,36
10,3578
10.3578
```

# Placeholders, concatenação e interpolação

```
int idade = 32;  
double saldo = 10.35784;  
String nome = "Maria";
```

```
Console.WriteLine("{0} tem {1} anos e tem saldo igual a {2:F2} reais", nome, idade, saldo);
```

```
Console.WriteLine($"{nome} tem {idade} anos e tem saldo igual a {saldo:F2} reais");
```

```
Console.WriteLine(nome + " tem " + idade + " anos e tem saldo igual a "  
    + saldo.ToString("F2", CultureInfo.InvariantCulture) + " reais");
```

 Microsoft Visual Studio Debug Console

```
Maria tem 32 anos e tem saldo igual a 10,36 reais  
Maria tem 32 anos e tem saldo igual a 10,36 reais  
Maria tem 32 anos e tem saldo igual a 10.36 reais
```

# Exercício de fixação

Em um novo programa, inicie as seguintes variáveis:

```
string produto1 = "Computador";  
string produto2 = "Mesa de escritório";  
  
byte idade = 30;  
int codigo = 5290;  
char genero = 'M';  
  
double preco1 = 2100.0;  
double preco2 = 650.50;  
double medida = 53.234567;
```

Em seguida, usando os valores das variáveis, produza a seguinte saída na tela do console:

```
Produtos:  
Computador, cujo preço é $ 2100,00  
Mesa de escritório, cujo preço é $ 650,50  
  
Registro: 30 anos de idade, código 5290 e gênero: M  
  
Medida com oito casas decimais: 53,23456700  
Arredondado (três casas decimais): 53,235  
Separador decimal invariant culture: 53.235
```

# Perguntas ??

