Fundamentos de C#

Datas, Horas, Números e Formatação de Dados

Softblue
cursos online

Tópicos Abordados

- Datas/Horas
  - A estrutura *DateTime*
  - Formatação
  - Operações com datas e horas
  - Intervalos de tempo com *TimeSpan*
- Geração de números randômicos
- Conversão de dados com a classe *Convert*
- A classe *Console*
  - Formatação de textos, números e alinhamento
  - Mais detalhes sobre a classe

Representação de uma Data/Hora

- A estrutura **System.DateTime** é utilizada para representar uma data/hora específica

```
DateTime data1 = new DateTime(2040, 3, 1);
```

ano, mês, dia

```
DateTime data2 = new DateTime(2040, 3, 1, 15, 30, 20);
```

ano, mês, dia, hora, minuto, segundo

```
DateTime data3 = DateTime.Parse("01/03/2040 15:30:20");
```

Cria um *DateTime* a partir de uma *string* (pode lançar *FormatException*)

Properties *Now* e *Today*

- *DateTime* possui as properties **Now** e **Today**, que retornam a data/hora atual do sistema

```
DateTime data = DateTime.Now;
DateTime data = DateTime.Today;
```

Today leva em consideração apenas a data

Formatação de Data/Hora

- O método *ToString()* de *DateTime* pode ser utilizado para representar uma data/hora como uma *string*

```
string format = data.ToString("d");
string format = data.ToString("t");
string format = data.ToString("T");
```

Formato DD/MM/AAAA

Formato HH:MM

Formato HH:MM:SS

- Para mais detalhes sobre a formatação de datas, consulte a documentação do C#

Operações com Datas

- *DateTime* possui alguns métodos para realizar operações com datas
  - *AddDays()*
  - *AddHours()*
  - *AddYears()*
  - etc.

```
DateTime data1 = DateTime.Today;
DateTime data2 = data1.AddYears(1);
```

Adiciona 1 ano à data atual

Se o valor for negativo, subtrai

Intervalos de Tempo

- A estrutura **TimeSpan** é utilizada para representar um intervalo de tempo
  - O intervalo é expresso no máximo em dias

```
TimeSpan delta = new TimeSpan(5, 30, 0);
```

Intervalo de 5 horas e 30 minutos

- Um *TimeSpan* pode ser usado para cálculo de datas

```
DateTime data1 = DateTime.Now;
DateTime data2 = data1.Add(delta);
```

Soma 5h30min na data/hora atual

```
DateTime data1 = DateTime.Now;
DateTime data2 = data1.Subtract(delta);
```

Subtrai 5h30min da data/hora atual

Números Randômicos

- A classe **System.Random** é utilizada na geração de números randômicos

```
Random r = new Random();
```

```
Random r = new Random(1000);
```

Semente

Uma sequência de números randômicos baseados na mesma semente é sempre igual

Geração de Números Randômicos

- *Random* possui vários métodos que permitem a geração de números randômicos

`int n = r.Next();`	0 <= *n* < *Int32.MaxValue*
`int n = r.Next(max);`	0 <= *n* < *max*
`int n = r.Next(min, max);`	min <= *n* < *max*
`double n = r.NextDouble();`	0.0 <= *n* < 1.0

Conversão de Dados

- C# possui a classe **System.Convert** para auxiliar na conversão de dados
  - Classe estática (apenas métodos estáticos)

```
int n = Convert.ToInt32("1111", 2);
```
Converte uma *string* no formato binário para *int*

```
char c = Convert.ToChar("A");
```
Converte uma *string* de 1 caractere para *char*

```
string s = Convert.ToString(50.2);
```
Converte um *double* para *string*

- Consulte a documentação da classe para conhecer mais métodos

A Classe *Console*

- A classe *Console* possui métodos estáticos que permitem enviar e receber dados do console (I/O)

```
string s = Console.ReadLine();
```
Lê uma linha digitada no console

```
Console.Write(s);
```
Escreve um texto no console

```
Console.WriteLine(s);
```
Escreve um texto no console e adiciona uma quebra no final

Formatação de Textos

- Existem situações onde é necessário juntar várias strings

```
int atual = 7, total = 14;
double perc = ((double)atual / total) * 100;
```

```
Console.WriteLine("Foram processados " + atual
    + " arquivos de " + total + " (" + perc + "%)");
```

Foram processados 7 arquivos de 14 (50%)

```
Console.WriteLine("Foram processados {0} arquivos de
    {1} ({2}%)", atual, total, perc);
```

Foram processados 7 arquivos de 14 (50%)

Não existe concatenação através do operador "+"

São usados *placeholders*

Alinhamento de Textos

- Um *placeholder* também pode ter informações a respeito de alinhamento

```
Console.WriteLine("{0}", "".PadLeft(43, '-'));
Console.WriteLine("| {0,-15} | {1,-12} | {2,-6} |", "Nome", "Cidade", "Idade");
Console.WriteLine("{0}", "".PadLeft(43, '-'));
Console.WriteLine("| {0,-15} | {1,-12} | {2,6} |", "José", "São Paulo", 35);
Console.WriteLine("| {0,-15} | {1,-12} | {2,6} |", "Pedro", "Manaus", 15);
Console.WriteLine("| {0,-15} | {1,-12} | {2,6} |", "Maria", "Natal", 60);
Console.WriteLine("{0}", "".PadLeft(43, '-'));
```

```
-------------------------------------------
| Nome            | Cidade       | Idade  |
-------------------------------------------
| José            | São Paulo    |     35 |
| Pedro           | Manaus       |     15 |
| Maria           | Natal        |     60 |
-------------------------------------------
```

Formatação de Números

- Além de indicar a posição do parâmetro, o *placeholder* pode ter mais informações, que auxiliam na formatação de números

Format String	Descrição
C ou **c**	Moeda
D ou **d**	Número inteiro
E ou **e**	Número exponencial
F ou **f**	Número decimal
N ou **n**	Formato numérico (com separadores)
P ou **p**	Porcentagem
X ou **x**	Hexadecimal

Exemplos de Formatação de Números

`Console.WriteLine("{0:C}", 10.35);`	R$ 10,35
`Console.WriteLine("{0:D8}", 36);`	00000036
`Console.WriteLine("{0:E1}", 1500);`	1,5E+003
`Console.WriteLine("{0:F2}", 32.5877);`	32,59
`Console.WriteLine("{0:N2}", 7430.968);`	7.430,97
`Console.WriteLine("{0:P1}", 0.68);`	68,0%
`Console.WriteLine("{0:X}", 15482);`	3C7A

Formatos Customizados

- É possível ainda definir formatos customizados para números

```
Console.WriteLine("{0:0000}", 25);
```
0025

```
Console.WriteLine("{0,8:000}", 6);
```
006

```
Console.WriteLine("{0:#C;#D;Zero}", 28);
Console.WriteLine("{0:#C;#D;Zero}", -28);
Console.WriteLine("{0:#C;#D;Zero}", 0);
```
28C
28D
Zero

```
Console.WriteLine("{0:###\\.###\\.###-##}",
    94829004721);
```
948.290.047-21

```
Console.WriteLine("{0:#####-###}", 57298900);
```
57298-900

O Método *String.Format()*

- Dados formatados podem ser armazenados em outra *string*, ao invés de irem para o console
- O método **String.Format()** permite isso

```
string s = String.Format("{0:D8}", 36);
```

Os dados formatados são retornados em uma nova *string*

Mais Detalhes da Classe *Console*

- Alterar as cores de frente e fundo

```
Console.ForegroundColor = ConsoleColor.Yellow;
Console.BackgroundColor = ConsoleColor.Blue;
```

- Limpar a tela

```
Console.Clear();
```

- Mudar o título da janela

```
Console.Title = "Minha Aplicação";
```

- Controlar a visibilidade do cursor

```
Console.CursorVisible = false;
```

Mais Detalhes da Classe *Console*

- Definir a posição do cursor

```
Console.CursorTop = 10;
Console.CursorLeft = 20;
```

- Ler um caractere digitado

```
ConsoleKeyInfo cki = Console.ReadKey(false);
char key = cki.KeyChar;
```

Indica se o caractere será mostrado na tela