



# Programação e Sistemas de Informação

CURSO PROFISSIONAL TÉCNICO DE  
GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

Estruturas de Dados Estáticas: **Strings**

**MÓDULO 4**

Professor: João Martiniano

# Introdução

- Neste módulo iremos analisar em detalhe dois componentes importantes da linguagem C#:
  - o tipo de dados `string`
  - arrays

# Introdução

- Uma string é um conjunto de caracteres
- As strings são delimitadas com aspas ( " )

# Declaração

- Existem diferentes formas de declarar uma string
- Exemplo 1: Declaração de string, sem inicialização

```
string nome;
```

- Exemplo 2: Declaração com string vazia

```
string morada1 = "";  
  
string morada2 = String.Empty;
```

- Exemplo 3: Declaração com valor literal

```
string cidade = "Coimbra";
```

# Testar string vazia

- Como testar se uma string está vazia?
- Existem várias formas, mas as mais simples são:

```
string nome = "";  
  
if (nome == "")  
{  
    Console.WriteLine("String vazia");  
}
```

```
string nome = "";  
  
if (nome == String.Empty)  
{  
    Console.WriteLine("*");  
}
```

# Anatomia de uma string

- Como já foi referido anteriormente, uma string é um conjunto de carateres
- Ou seja, é um conjunto de valores do tipo `char`
- Cada carater de uma string está posicionado num determinado **índice**
- O primeiro carater tem índice `0`
- Por exemplo a seguinte string...

```
string mensagem = "Hello World";
```

- ... é na realidade um conjunto de carateres:

H	e	l	l	o		W	o	r	l	d
---	---	---	---	---	--	---	---	---	---	---

Índice → 0 1 2 3 4 5 6 7 8 9 10

# Anatomia de uma string

- É possível aceder a cada carater de uma string, utilizando a seguinte sintaxe:

`variável[índice]`

- Exemplo 1: mostrar o primeiro carater da string `mensagem`

```
string mensagem = "Hello World";  
  
Console.WriteLine(mensagem[0]);
```

# Anatomia de uma string

- Não é possível, no entanto, alterar caracteres individuais de uma string
- Exemplo 2: tentar modificar o 2º carater da string `mensagem`

```
string mensagem = "Hello World";  
  
// A seguinte instrução irá provocar um erro  
mensagem[1] = 'E';
```



# Anatomia de uma string

- Exemplo 3: mostrar os caracteres da string `mensagem`, intercalados com o carater -

```
string mensagem = "Hello World";  
  
for (int i = 0; i < mensagem.Length; ++i)  
{  
    Console.WriteLine("{0}-", mensagem[i]);  
}
```

- Resultado:

```
H-e-l-l-o- -W-o-r-l-d-
```

# Anatomia de uma string

- Podemos atribuir a uma variável do tipo `char` um carater de uma string...

```
string mensagem = "Hello World";  
  
char a = mensagem[0];
```

- ... mas não a uma variável do tipo `string`:

```
string mensagem = "Hello World";  
  
// A seguinte instrução irá provocar um erro  
string a = mensagem[0];
```

# A classe `String`

- Na realidade, uma string é uma instância da classe `String`
- Ou seja, é um objeto com um conjunto de propriedades e métodos pré-definidos, os quais fornecem um conjunto de funcionalidades úteis
- Iremos analisar os seguintes métodos:
  - `Contains()`
  - `IndexOf()`
  - `LastIndexOf()`
  - `Insert()`
  - `Replace()`
  - `ToLower()`
  - `ToUpper()`
  - `Trim()`

## A classe `String`: A propriedade `Length`

- A propriedade `Length` retorna o número de carateres de uma string
- Exemplo 1:

```
string frase = "The quick brown fox jumped over the lazy dog";
```

```
Console.WriteLine("A frase '{0}' contém {1} carateres", frase,  
frase.Length);
```

```
A frase 'The quick brown fox jumped over the lazy dog' contém 44  
carateres
```

- Exemplo 2: Obter o último carater de uma string

```
string frase = "The quick brown fox jumped over the lazy dog";
```

```
char ultimoCarater = frase[frase.Length - 1];
```

## A classe `String`: Método `Contains()`

- O método `Contains()` verifica se uma string contém uma substring
- Retorna:
  - `true` se a string contiver a substring
  - `false` caso contrário
- Exemplo:

```
string frase = "The quick brown fox jumped over the lazy dog";  
  
if (frase.Contains("lazy dog"))  
{  
    Console.WriteLine("*");  
}
```

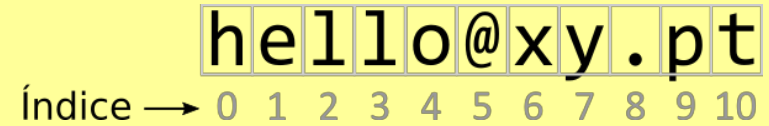
Verificar se a string `frase`  
contém a substring `"lazy dog"`

## A classe `String`: Método `IndexOf()`

- O método `IndexOf()` retorna o índice da **primeira** ocorrência de um carater ou de uma substring dentro de uma string
- Na realidade este método tem várias variantes que permitem muita flexibilidade neste tipo de teste
- Iremos apenas ver três variantes deste método:
  - variante 1: pesquisar um carater dentro de uma string
  - variante 2: pesquisar uma substring dentro de uma string
  - variante 3: pesquisar uma substring dentro de uma string, a partir de um determinado índice
- Este método retorna o valor `-1` quando não encontra o carater ou a substring especificada

## A classe `String`: Método `IndexOf()`

- No exemplo são efetuadas as seguintes pesquisas, dentro da string `email`:
  - o carater '@'
  - a substring `"pt"`
  - a substring `"hello"`, a partir do índice 5



h	e	l	l	o	@	x	y	.	p	t
0	1	2	3	4	5	6	7	8	9	10

Índice →

- Exemplo:

```
string email = "hello@xy.pt";
```

```
Console.WriteLine("Índice do carater '@': {0}", email.IndexOf('@'));  
Console.WriteLine("Índice da substring \"pt\": {0}", email.IndexOf("pt"));  
Console.WriteLine("Índice da substring \"hello\": {0}", email.IndexOf("hello", 5));
```

- Resultado:

```
Índice do carater '@': 5  
Índice da substring "pt": 9  
Índice da substring "hello": -1
```

## A classe `String`: Método `LastIndexOf()`

- O método `LastIndexOf()` retorna o índice da **última** ocorrência de um carater ou de uma substring dentro de uma string
- Este método tem várias variantes
- Iremos apenas ver duas variantes deste método:
  - variante 1: pesquisar a última ocorrência de um carater dentro de uma string
  - variante 2: pesquisar a última ocorrência de uma substring dentro de uma string
- Este método retorna o valor `-1` quando não encontra o carater ou a substring especificada



## A classe `String`: Método `LastIndexOf()`

- Dada a string `email...`

u	t	i	l	i	z	a	d	o	r	.	p	t	@	g	m	a	i	l	.	p	t
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Índice → 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

- ... e o seguinte código:

```
string email = "utilizador.pt@gmail.pt";  
  
Console.WriteLine("Índice do último carater '.': {0}", email.LastIndexOf('.'));  
Console.WriteLine("Índice da última substring \"pt\": {0}", email.LastIndexOf("pt"));
```

- Resultado:

```
Índice do último carater '.': 19  
Índice da última substring "pt": 20
```

## A classe `String`: Método `Insert()`

- O método `Insert()` permite inserir uma string dentro de outra string, a partir de um determinado índice
- O método retorna a nova string
- Exemplo:

```
string email = "utilizador@.com";  
  
Console.WriteLine("Email original: {0}", email);  
  
email = email.Insert(11, "gmail");  
  
Console.WriteLine("Novo email: {0}", email);
```

Inserir a string `"gmail"` a partir do índice 11

- Resultado:

```
Email original: utilizador@.com  
Novo email: utilizador@gmail.com
```

## A classe `String`: Método `Replace()`

- O método `Replace()` substitui todas as ocorrências de uma string ou de um carater
- Tem duas variantes:
  - variante 1: substitui todas as ocorrências de um carater, por outro carater
  - variante 2: substitui todas as ocorrências de uma string, por outra string
- Este método retorna uma nova string, com o resultado das substituições

# A classe `String`: Método `Replace()`

## Variante 1

- Substituir, no seguinte exemplo, todas as ocorrências do carater espaço, por vírgula

```
string numeros = "1 2 3 4 5 6 7 8 9";  
string numerosCsv = numeros.Replace(' ', ',');  
  
Console.WriteLine("Números (original): {0}", numeros);  
Console.WriteLine("Números (CSV): {0}", numerosCsv);
```

Substituir espaço  
por vírgula

- Resultado:

```
Números (original): 1 2 3 4 5 6 7 8 9  
Números (CSV): 1,2,3,4,5,6,7,8,9
```

# A classe `String`: Método `Replace()`

## Variante 2

- Substituir, no seguinte exemplo:
  - todas as ocorrências da string `<b>` pela string `<strong>`
  - todas as ocorrências da string `</b>` pela string `</strong>`

```
string html1 = "<b>The</b> quick brown <b>fox</b> jumped...";  
string html2
```

```
html2 = html1.Replace("<b>", "<strong>");
```

Substituir `<b>` por `<strong>`

```
html2 = html2.Replace("</b>", "</strong>");
```

Substituir `</b>` por `</strong>`

```
Console.WriteLine("HTML original: {0}", html1);  
Console.WriteLine("HTML modificado: {0}", html2);
```

- Resultado:

```
HTML original: <b>The</b> quick brown <b>fox</b> jumped...  
HTML modificado: <strong>The</strong> quick brown <strong>fox</strong>  
jumped...
```

## A classe `String`: Método `Replace()`

- É importante notar que o exemplo anterior poderia ser simplificando utilizando uma técnica chamada *method chaining*
- Ou seja, encadear vários métodos sucessivamente
- Os métodos são executados sequencialmente, da esquerda para a direita
- As seguintes linhas...

```
html2 = html1.Replace("<b>", "<strong>");  
html2 = html2.Replace("</b>", "</strong>");
```

- ... podem ser simplificadas com *method chaining*:

```
html2 = html1.Replace("<b>", "<strong>").Replace("</b>", "</strong>")
```

Substituir `<b>` por `<strong>`

Substituir depois `</b>` por  
`</strong>`

## A classe `String`: Métodos `ToLower()` e `ToUpper()`

- Os métodos `ToLower()` e `ToUpper()` convertem os caracteres de uma string, respetivamente, em caracteres minúsculos e maiúsculos
- Exemplo:

```
string frase = "The Quick Brown Fox Jumped Over The Lazy Dog";  
string minusculas = frase.ToLower();  
string maiusculas = frase.ToUpper();  
  
Console.WriteLine("Original: {0}", frase);  
Console.WriteLine("ToLower(): {0}", minusculas);  
Console.WriteLine("ToUpper(): {0}", maiusculas);
```

- Resultado:

```
Original: The Quick Brown Fox Jumped Over The Lazy Dog  
ToLower(): the quick brown fox jumped over the lazy dog  
ToUpper(): THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG
```

## A classe `String`: Método `Trim()`

- O método `Trim()` retira espaços no início e no fim de uma string
- Exemplo:

```
string username = " fjp1000 ";  
  
Console.WriteLine("String original: \"{0}\"", username);  
Console.WriteLine("Com método Trim(): \"{0}\"", username.Trim());
```

- Resultado:

```
String original: " fjp1000 "  
Com método Trim(): "fjp1000"
```



## A classe `String`: Método `Substring()`

- O método `Substring()` retorna uma parte de uma string
- Ou seja, retorna uma substring a partir de uma string
- Existem duas formas de utilizar este método:

`Substring(índice)`

Retorna o conteúdo de uma string a partir de um determinado índice

`Substring(índice, comprimento)`

Retorna uma substring a partir de um determinado índice, com um determinado comprimento

## A classe `String`: Método `Substring()`

- Exemplo:

Índice →

u	s	e	r	@	g	m	a	i	l	.	p	t
0	1	2	3	4	5	6	7	8	9	10	11	12

u	s	e	r	@	g	m	a	i	l	.	p	t
---	---	---	---	---	---	---	---	---	---	---	---	---

→

`Substring(5)`

Retorna todos os caracteres a partir do índice 5

u	s	e	r	@	g	m	a	i	l	.	p	t
---	---	---	---	---	---	---	---	---	---	---	---	---

→

`Substring(5, 5)`

Retorna 5 caracteres a partir do índice 5

```
string email = "user@gmail.pt";

string s1 = email.Substring(5);
string s2 = email.Substring(5, 5);
```

# A classe `String`

- É importante realçar que os métodos e propriedades podem ser aplicados a valores literais
- Por exemplo:

```
Console.WriteLine("{0}", "1 2 3 4 5 6 7 8 9".Replace(" ", ","));
```

```
Console.WriteLine("A string tem {0} carateres", "abc".Length);
```

## ***String interpolation***

- Na versão 6.0 da linguagem C# foi introduzido um novo modelo de definição de strings: as *interpolated strings*
- São strings que:
  - começam com o carater `$`
  - contêm *interpolation expressions* as quais são substituídas pela sua representação em string
- Uma *interpolation expression* é delimitada pelos caracteres `{ }`
- As strings criadas com *interpolation expressions* são mais fáceis de criar e ler (pelo programador)

## String interpolation

- Por exemplo, tradicionalmente para colocar numa string o valor de duas variáveis:

```
int dia = 12;  
double graus = 16.7;  
  
string mensagem = "Hoje é dia " + dia + " e estão " + graus + " graus  
celsius.";
```

- Utilizando *string interpolation* é mais fácil inicializar a variável `mensagem`:

```
int dia = 12;  
double graus = 16.7;  
  
string mensagem = $"Hoje é dia {dia} e estão {graus} graus celsius.";
```

Importante!

Variável `dia`

Variável `graus`

## String interpolation

- Para mostrar os valores das variáveis na consola, com o método tradicional (ou seja, utilizando *placeholders*):

```
int dia = 12;  
double graus = 16.7;  
  
Console.WriteLine("Hoje é dia {0} e estão {1} graus celsius.", dia,  
graus);
```

- Utilizando *string interpolation*:

```
int dia = 12;  
double graus = 16.7;  
  
Console.WriteLine($"Hoje é dia {dia} e estão {graus} graus celsius.");
```

Importante!

Variável `dia`

Variável `graus`

## *String interpolation*

- Para incluir os caracteres { } numa string interpolada, deve ser utilizada a sequência {{ e }}:

```
string mensagem = $"Uma interpolation expression começa com o carater {{  
e termina com o carater }}";
```