

VIANNA JUNIOR
INSTITUTO



Algoritmos

Strings

Professor: Camillo Falcão

Sequências de Caracteres

- Sequências de caracteres justapostos são fundamentais no desenvolvimento de programas computacionais.
- Exemplos de sequências de caracteres (representadas internamente num programa):
 - Mensagem de e-mail;
 - Texto de um programa;
 - Nome e endereço em cadastro de clientes, alunos;
 - Sequencia genética. Um gene (ou o DNA de algum organismo) é representado por sequencias dos caracteres A, T, G e C (nucleotídeos);
 - E etc...

Caracteres em C#

- Os caracteres em C# são representados internamente por códigos numéricos (Unicode);

Alguns caracteres visíveis (podem ser impressos)

	0	1	2	3	4	5	6	7	8	9
30			sp	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}				

sp: espaço em branco

Sequências de Caracteres

- Uma variável usada para armazenar um caractere é representada da seguinte maneira:

```
char letra; // variavel letra do tipo character
letra = 'a'; // atribuida a letra "a" para a variavel
```

- Se em uma variável do tipo **char** podemos armazenar somente um caractere, então para armazenar vários caracteres (ex: “jose”, “carro”) é necessário **utilizar as sequências de caracteres**, representadas pela classe **String**.
- Em Programação, sequências de caracteres são usualmente chamadas de **strings**.

Strings

- Exemplo de declaração:

```
string cidade;
```

- A variável **cidade** é uma cadeia de caracteres e pode armazenar qualquer cadeia de caracteres.

	0	1	2	3	4	5	6	7	8	9	10	11
cidade	J	u	i	z		d	e		F	o	r	a

- Outras formas de inicializarmos *strings* em C#:

```
string cidade = "Rio";  
string disc = "Algoritmo";
```

Strings: Manipulação

- Como uma string é um conjunto de caracteres, podemos manipular os seus caracteres.
- Exemplo:

```
string nome = "Algoritmos";  
  
char inicial = nome[0];  
Console.WriteLine("Primeira letra: {0}\n", inicial);
```

Strings: Entrada e Saída

```
void main(string[] args)
{
    string s;
    Console.Write("Digite uma string: ");
    s = Console.ReadLine();
    Console.Write("String digitada: {0}", s);
}
```


Strings - Exemplo 1

- O programa a seguir imprime uma *string*, caractere por caractere:

```
void main(string[] args)
{
    string s;
    int i;
    Console.Write("Digite uma string: ");
    s = Console.ReadLine();

    //Imprime cada caractere da string lida
    for(i=0; i < s.Length ; i++)
        Console.Write("{0}", s[i]);
}
```

- Note que, o **for** acima equivale a `Console.Write("{0}", s);`

Exercício resolvido 1

- Criar uma função que receba como parâmetro uma string (*cadeia*), e caractere (*procurado*). A função deverá retornar a quantidade de vezes que o caractere *procurado* foi encontrado na *cadeia*.
- Solução proposta:
 - Precisamos “varrer” a cadeia de caracteres (estrutura de repetição) e contar quantos são iguais ao caractere procurado, caractere a caractere.

Exercício 1 – Solução proposta

```
int conta(string str, char procurado)
{
    int cont, i;
    i = 0;
    cont = 0;
    while (i < str.Length)
    {
        if (str[i] == procurado)
        {
            cont++;
        }
        i++;
    }
    return cont;
}
```

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i =
cont =

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4  int cont, i;  
5  i = 0;  
6  cont = 0;  
7  
8  while ( i < str.Length )  
9  {  
10     if ( str[i] == procurado )  
11     {  
12         cont++;  
13     }  
14     i++;  
15 }  
16 return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = ?
cont = ?

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 0
cont = 0

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"

str.Length = 5

procurado = 't'

Variáveis:

i = 0

cont = 0

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 0
cont = 0

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 0
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 1
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"

str.Length = 5

procurado = 't'

Variáveis:

i = 1

cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 1
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 2
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"

str.Length = 5

procurado = 't'

Variáveis:

i = 2

cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 2
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 3
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"

str.Length = 5

procurado = 't'

Variáveis:

i = 3

cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 3
cont = 1

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 3
cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 4
cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"

str.Length = 5

procurado = 't'

Variáveis:

i = 4

cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 4
cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 5
cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"

str.Length = 5

procurado = 't'

Variáveis:

i = 5

cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício 1 – Teste de Mesa

```
1  int conta( string str,  
2           char procurado )  
3  {  
4      int cont, i;  
5      i = 0;  
6      cont = 0;  
7  
8      while ( i < str.Length )  
9      {  
10         if ( str[i] == procurado )  
11         {  
12             cont++;  
13         }  
14         i++;  
15     }  
16     return cont;  
17 }
```

Entrada:

str = "teste"
str.Length = 5
procurado = 't'

Variáveis:

i = 5
cont = 2

i	0	1	2	3	4
str	t	e	s	t	e

Exercício resolvido 2

- Criar uma função para verificar se a string *s2* está contida na *string s1*. A função deverá retornar *true* se encontrar a *string* ou *false*, caso contrário.
- Exemplo:
 - Se *s1* fosse “*Ana Maria Silva*” e *s2* fosse “*Maria*”, a função retornaria *true*, pois *s2* está contido em *s1*.

Exercício 2 – Solução proposta

```
bool BuscarString(string s1, string s2)
{
    int i, j, aux, tam1, tam2;
    tam1 = s1.Length;
    tam2 = s2.Length;
    for(i=0; i<tam1; i++)
    {
        aux=i;
        for(j=0; j<tam2 && aux<tam1; j++)
        {
            if (s2[j] != s1[aux])
                break;
            aux++;
        }
        if (j == tam2)
            return true;
    }
    return false;
}
```

Exercício 2 – Debugging

Crie o método principal (Main) e leia duas strings:

s1 = "Este é um teste"

s2 = "um"

Insira um "Break Point" na linha tam = s1.Length e rode o sistema em modo de depuração. A cada linha executada, confira os valores das variáveis i, j e aux;

Exercício 2 – Completo

```
bool BuscarString(string s1, string s2)
{
    int i, j, aux, tam1, tam2;
    tam1 = s1.Length;
    tam2 = s2.Length;
    for(i=0; i<tam1; i++)
    {
        aux=i;
        for(j=0; j<tam2 && aux<tam1; j++)
        {
            if (s2[j] != s1[aux])
                break;
            aux++;
        }
        if (j == tam2)
            return true;
    }
    return false;
}
```

```
void Main(string[] args)
{
    string s1, s2;
    bool res;
    s1 = Console.ReadLine();
    s2 = Console.ReadLine();
    res = BuscarString(s1, s2);
    if(res)
        Console.Write("Encontrou");
    else
        Console.Write("Nao
encontrou");
}
```

Exercícios

- 1) Fazer um programa para contar o número de espaços em branco de uma *string*.
- 2) Refaça o programa anterior criando uma função que receberá como parâmetro a *string* e retornará o número de espaços em branco que a *string* contém.
- 3) Fazer um programa para contar o número de vogais em uma *string*.
- 4) Refaça o programa anterior criando uma função que receberá como parâmetro a *string* e retornará o número de vogais que a *string* contem.

Exercícios

- 5) Escreva um programa para ler uma *string* (com mais de uma palavra) e imprimi-la de forma que a primeira letra de cada palavra fique em maiúscula. Para isso, basta subtrair 32 do elemento que deseja alterar para maiúsculo.

```
Console.Write("{0} ", Convert.ToChar(chrNomeRecebido[i] - 32));
```

ou

```
Console.Write("{0} ", Convert.ToChar(chrNomeRecebido[i] - 'a' + 'A' ));
```

Exemplo:

Entrada: lab. de linguagem de programacao

Impressão: Lab. De Linguagem De Programacao

Exercícios

- 6) Crie um procedimento que receba uma frase e a exiba na tela de forma soletrada, ou seja, cada letra deve ser exibida na tela separada por hífen.
- 7) Crie um procedimento que receba uma *string* e imprima a string invertida.

Exemplo:

Entrada: Teste

Saída: etseT

Aula de Exercícios

Concatenando *Strings*

- Em C#, é possível concatenar ***strings*** utilizando para isso o operador **+**, como no exemplo abaixo.
- Exemplos:

```
string str1 = "Fundamentos";  
string str2 = "de";  
string str3 = "Programação";  
string str4;  
  
str4 = str1 + ' ' + str2 + ' ' + str3;
```

Igualdade entre *Strings*

- É possível utilizar o operador `==` para testar a igualdade entre strings em C#. Veja o exemplo abaixo:

```
string s1;  
string s2;  
s1 = "Algoritmos";  
s2 = "Algo" + "ritmos";  
Console.Write(s1 == s2); //Imprime true
```

Strings

- Sub-rotina com *string*

```
int Contar(string str,
           char procurado)
{
    int cont, i;
    i = 0;
    cont = 0;
    while (i < str.Length)
    {
        if (str[i] == procurado)
        {
            cont++;
        }
        i++;
    }
    return cont;
}
```

```
void Main(string[] args)
{
    string nome = "Vianna";
    int total;
    total = Contar(nome, 'a');
}
```

A classe *StringBuilder*

- A classe `StringBuilder` permite a concatenação de strings sem o comprometimento de desempenho do operador `+`.

```
StringBuilder sb = new StringBuilder();  
sb.Append('A');  
sb.Append('l');  
sb.Append("goritmos");  
Console.WriteLine("{0}", sb.ToString()); //Imprime Algoritmos
```

Exercícios

- 1) Faça um programa que leia duas *strings* e crie uma terceira *string* que será a concatenação das duas outras *strings* lidas.
- 2) Fazer um procedimento para imprimir uma *string* recebida como parâmetro sem os espaços em branco.
- 3) Fazer um procedimento para receber uma *string* do usuário e imprimir uma estatística dos caracteres digitados. Isto é, imprimir o número de vogais, consoantes e outros caracteres.
- 4) Fazer um programa para ler uma *string* e transferir as consoantes para uma *string* e as vogais para outra. Depois mostre cada uma das *strings*.

Exercícios

- 5) Faça uma função que receba uma *string* do usuário (máx. 20 caracteres) e um caractere qualquer. A função deverá criar uma nova string onde terão sido removidas todas as ocorrências do caractere da *string*. A função deve retornar o número de remoções.

- 6) Escreva uma função que receba uma cadeia de caracteres de tamanho máximo 100, e retorne *true* se esta cadeia é uma palíndrome e *false* caso contrário. Uma palavra é dita ser palíndrome se a sequência de seus caracteres da esquerda para a direita é igual à sequência de seus caracteres da direita para a esquerda. Ex.: **arara**, **asa**.

Exercícios

- 7) Um dos sistemas de encriptação mais antigos é atribuído a Júlio César: se uma letra a ser encriptada é a letra de número N do alfabeto, substitua-a com a letra $(N+K)$, onde K é um número inteiro constante (César utilizava $K = 3$).

Dessa forma, para $K = 1$ a mensagem “Ataque ao amanhecer” se torna “bubrfabpabnboifdfs”. Faça um programa que receba como entrada uma mensagem e um valor de K e retorne a mensagem criptografada pelo código de César.