



# **Algoritmos e Linguagem de Programação**

## **Capitulo 7**

### **Programação com Métodos**

Prof. Me. Renato Carioca Duarte



# Métodos

- **Um método é um trecho de código de programa independente de qualquer parte do programa**, mas relacionado ao programa com atribuições bem definidas.
- Os métodos são um conjunto de instruções que efetuam uma tarefa específica. Os métodos também podem ser chamados de sub-rotinas.
- De forma geral, **os métodos podem receber valores de entrada (parâmetros opcionais) e gerar opcionalmente um valor de saída (retornar apenas um valor), denominado valor de retorno.**



# Métodos

- A utilização de métodos torna o trabalho de desenvolvimento com linguagem C# algo bastante versátil, já que:
  - Em termos de modularidade, tem-se o programa dividido em vários módulos (divisão e conquista), e cada módulo desempenha uma ação particular. **Essa estratégia de programação facilita a manutenção dos programas construídos.**
  - O programador torna-se capaz de criar seus próprios pacotes de métodos pessoais, fazendo com que a programação se torne mais eficiente, **porque é possível aproveitar códigos de programas que já foram testados e revisados anteriormente**, os quais podem ser usados sem problema em novos programas.
- No geral, problemas complexos exigem algoritmos complexos, mas **sempre é possível dividir um problema grande em problemas menores.** Dessa forma, cada parte menor tem um algoritmo mais simples, e é esse trecho menor que na linguagem C# pode se tornar um método.



# Definição de Métodos

- Um método externo definido pelo programador ocorre conforme a seguinte sintaxe:

```
<qualificador> <tipo> <nome> ([Parâmetros])  
  
{  
  
    <corpo>;  
  
}
```

< qualificador > pode ser:

- **private** (quando pode ser acessado dentro da classe a que pertence),
- **public** (quando pode ser acessado fora da classe a que pertence),
- **protected** (quando somente pode ser acessado pela classe a que pertence ou por suas classes-filho que estejam herdando as características de comportamento da classe-pai).
- Segunda cláusula opcional denominada **static** que tem por finalidade indicar que o método estático pertence ao próprio tipo, em vez de um objeto específico



# Definição de Métodos

```
<qualificador> <tipo> <nome> ([Parâmetros])  
{  
    <corpo>;  
}
```

- < tipo >: é o **tipo de dado de retorno do método após sua execução**. Pode ser o nome de um tipo primitivo da linguagem como: **int**, **float**, **double**, **string**, long, byte, entre outros. Se o método não for retornar valor, deve-se usar a palavra reservada **void**.
- <nome>: é o nome com o qual o método será chamado.
- <parâmetros>: um ou mais elementos opcionais com os quais o método efetua a **entrada de valores para sua operação**. Caso o método tenha mais de um parâmetro, separe-os com virgula.
- <corpo>: é o código que executa a ação do método.




# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```

A dark blue rectangular box representing a terminal window. It contains the text 'Main - Linha 1' in a light blue monospaced font, which is the first line of output from the provided code.



# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```

A dark blue rectangular box representing a terminal window. It contains two lines of white text: "Main - Linha 1" on the first line and "Main - Linha 2" on the second line.

```
Main - Linha 1  
Main - Linha 2
```



# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    → Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```

A dark blue rectangular box representing a terminal window. It contains two lines of white text: "Main - Linha 1" on the first line and "Main - Linha 2" on the second line.

```
Main - Linha 1  
Main - Linha 2
```





# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
Main - Linha 1  
Main - Linha 2  
Exemplo - Linha 1
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```



# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```

```
Main - Linha 1  
Main - Linha 2  
Exemplo - Linha 1  
Exemplo - Linha 2
```



# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```

```
Main - Linha 1  
Main - Linha 2  
Exemplo - Linha 1  
Exemplo - Linha 2  
Exemplo - Linha 3
```



# Uso de Métodos Sem Retorno

- Quando um método é chamado o fluxo de execução do programa, é desviado para o método em questão e, após executar o método, o fluxo do programa retorna para o ponto de chamada para dar continuidade à execução do programa.

```
public static void Main (string[] args) {  
    Console.WriteLine ("Main - Linha 1");  
    Console.WriteLine ("Main - Linha 2");  
    Exemplo ();  
    Console.WriteLine ("Main - Linha 3");  
}
```

```
public static void Exemplo ()  
{  
    Console.WriteLine ("Exemplo - Linha 1");  
    Console.WriteLine ("Exemplo - Linha 2");  
    Console.WriteLine ("Exemplo - Linha 3");  
}
```

```
Main - Linha 1  
Main - Linha 2  
Exemplo - Linha 1  
Exemplo - Linha 2  
Exemplo - Linha 3  
Main - Linha 3
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

A dark-themed terminal window showing the output of the program. The first line of output is "Main - Linha 1".

Main - Linha 1



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    → MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
```





# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
OutroMetodo - Linha 1
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
OutroMetodo - Linha 1
OutroMetodo - Linha 2
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
OutroMetodo - Linha 1
OutroMetodo - Linha 2
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
OutroMetodo - Linha 1
OutroMetodo - Linha 2
MetodoXX - Linha 2
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
OutroMetodo - Linha 1
OutroMetodo - Linha 2
MetodoXX - Linha 2
Main - Linha 3
```



# Método Chamando Métodos

```
public static void Main (string[] args)
{
    Console.WriteLine ("Main - Linha 1");
    Console.WriteLine ("Main - Linha 2");
    MetodoXX ();
    Console.WriteLine ("Main - Linha 3");
}
public static void MetodoXX ()
{
    Console.WriteLine ("MetodoXX - Linha 1");
    OutroMetodo ();
    Console.WriteLine ("MetodoXX - Linha 2");
}
public static void OutroMetodo ()
{
    Console.WriteLine ("OutroMetodo - Linha 1");
    Console.WriteLine ("OutroMetodo - Linha 2");
}
```

```
Main - Linha 1
Main - Linha 2
MetodoXX - Linha 1
OutroMetodo - Linha 1
OutroMetodo - Linha 2
MetodoXX - Linha 2
Main - Linha 3
```



# Passagem de Parâmetros

- O uso de passagem de parâmetros em um método proporciona uma comunicação bidirecional entre as rotinas do programa.
- As variáveis locais só ocupam espaço em memória quando estão sendo utilizadas.
- Quando se trabalha com métodos, é possível passar valores de uma sub-rotina para outra.
- Dessa forma, pode-se passar valores de uma sub-rotina ou rotina chamadora a outra sub-rotina e vice-versa.
- A passagem de parâmetro entre métodos poderá ser definida sob a ótica de dois comportamentos:
  - passagem de parâmetro por valor e
  - passagem de parâmetro por referência





# Passagem de parâmetro por Valor

- Usa-se a passagem de parâmetro por valor quando há necessidade de passar algum conteúdo à sub-rotina chamada.
- O conteúdo passado pelo parâmetro na chamada da sub-rotina é copiado a partir da chamada para a sub-rotina chamada.
- Qualquer modificação na variável local da sub-rotina não afetará o valor do parâmetro passado, ou seja, o processamento é executado apenas na sub-rotina, ficando assim o resultado obtido “preso” na área de abrangência da sub-rotina chamada.



# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

a

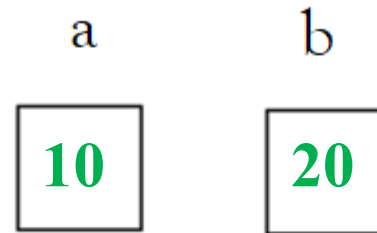
10

```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```



# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```



# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

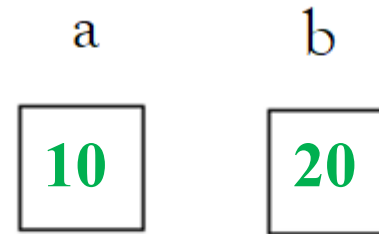


```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```

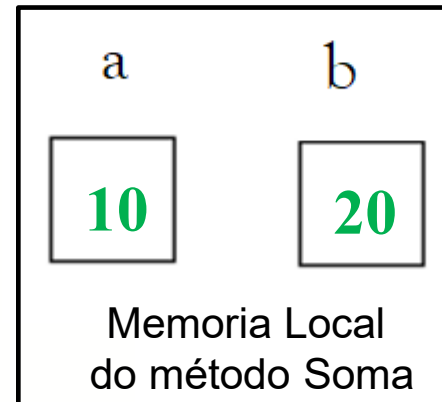


# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
➡ public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```



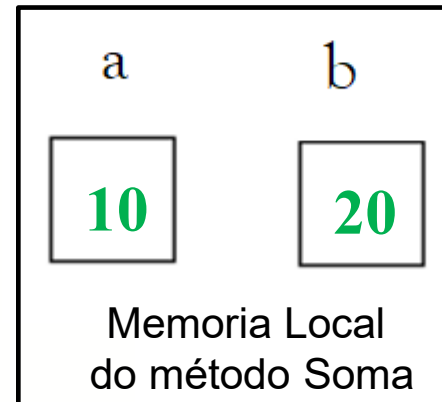


# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```



30

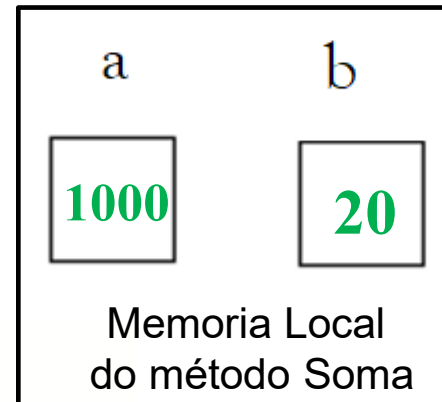


# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```

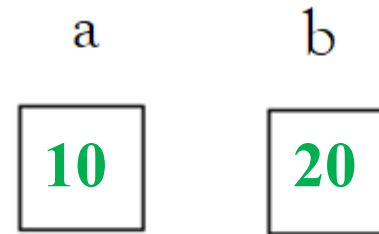


30

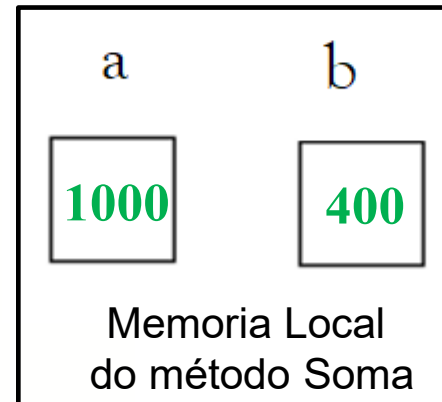


# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```



30



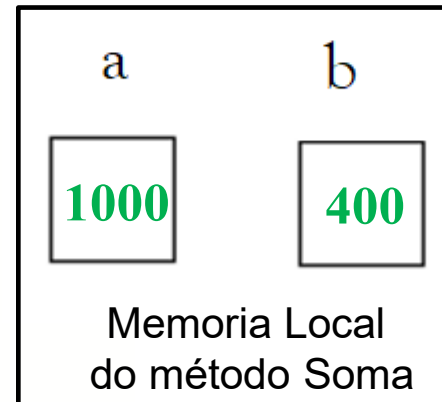


# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```



30  
1400

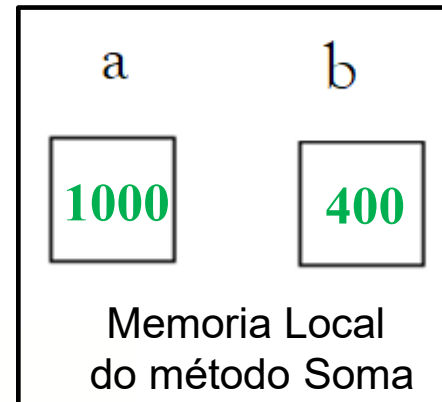


# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```

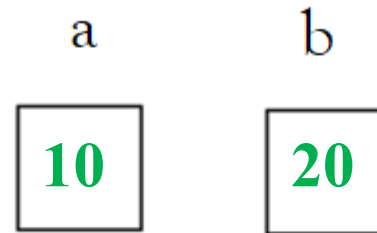


30  
1400



# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```

```
30  
1400  
10
```



# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



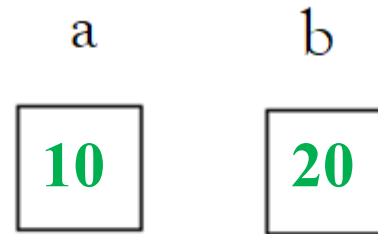
```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```

```
30  
1400  
10  
20
```



# Passagem de parâmetro por Valor

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    Soma(a,b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void Soma(int a, int b)  
{  
    Console.WriteLine (a + b);  
    a = a * 100;  
    b = b * 20;  
    Console.WriteLine (a + b);  
}
```

```
30  
1400  
10  
20
```



# Passagem de parâmetro por Referência

- A passagem de parâmetro por referência ocorre quando uma sub-rotina envia um conteúdo para outra sub-rotina e **aguarda um retorno nesse parâmetro com um conteúdo processado**.

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Passagem de parâmetro por Referência

- A passagem de parâmetro por referência ocorre quando uma sub-rotina envia um conteúdo para outra sub-rotina e **aguarda um retorno nesse parâmetro com um conteúdo processado**.

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

a

10

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```





# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```



```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

a

b

10

20

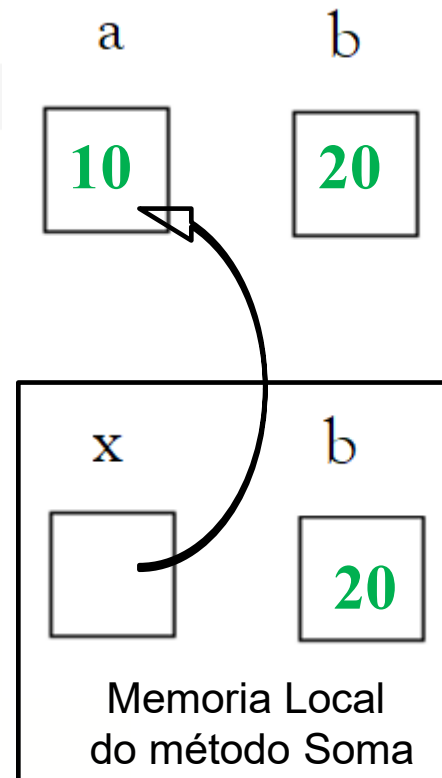
```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
➡ public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```

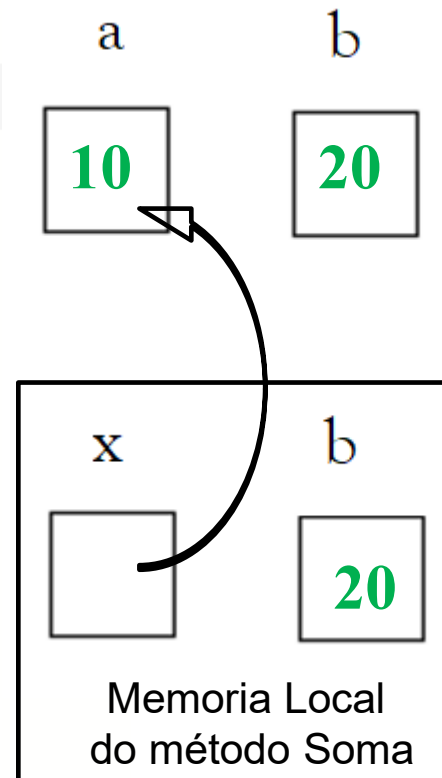




# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



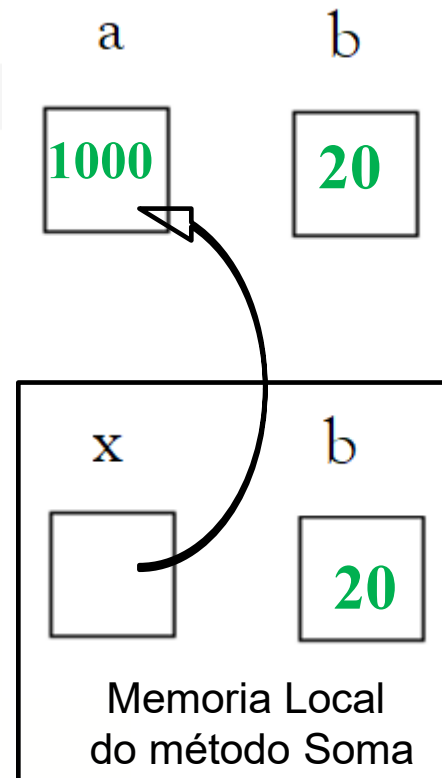
30



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



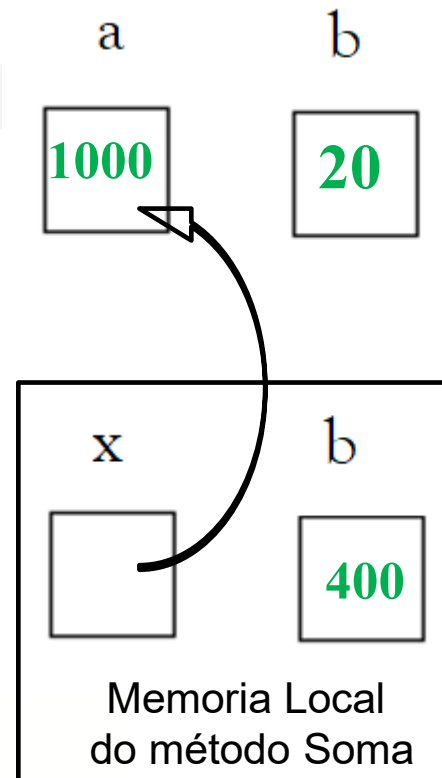
30



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



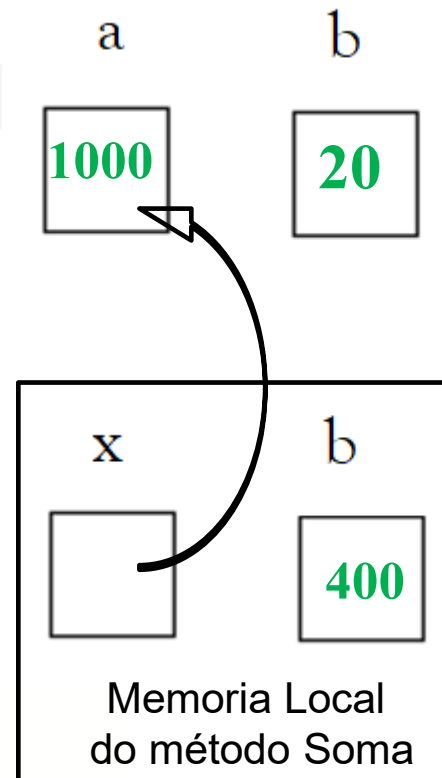
30



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



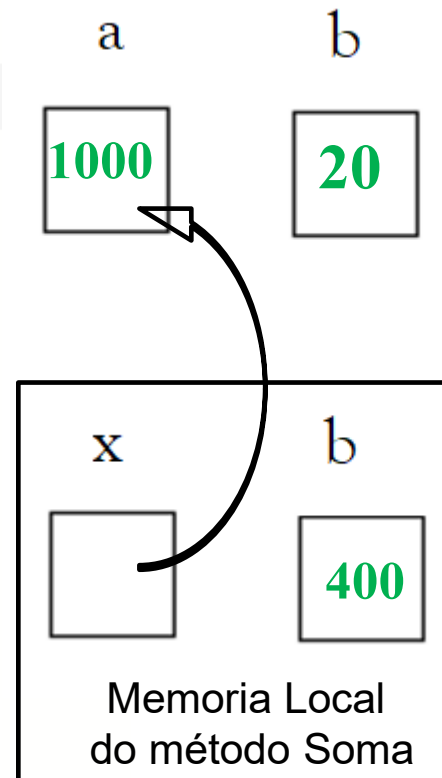
30  
1400



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



30  
1400





# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

a

b

1000

20

30  
1400  
1000

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

a

b

1000

20

30  
1400  
1000  
20

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Passagem de parâmetro por Referência

```
public static void Main (string[] args) {  
    int a = 10;  
    int b = 20;  
    S (ref a, b);  
    Console.WriteLine (a);  
    Console.WriteLine (b);  
}
```

a

1000

b

20

30  
1400  
1000  
20

```
public static void S(ref int x,int b)  
{  
    Console.WriteLine (x + b);  
    x = x * 100;  
    b = b * 20;  
    Console.WriteLine (x + b);  
}
```



# Uso de Método com Retorno

- Um método com retorno é uma estrutura que possui como característica operacional a capacidade de sempre **retornar um conteúdo como resposta**, independentemente de fazer uso de passagem de parâmetros por valor ou por referência.
- Para cumprir o requisito de sempre retornar uma resposta uma sub-rotina do tipo função, **faz uso no final de um comando return com algum conteúdo de resposta associado a função**.
- Uma típica sub-rotina de função tem por finalidade, sempre, retornar um conteúdo como resposta, e para tal feito deve ser definida com um dos tipos de dados primitivos da linguagem.



# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```

```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```

a

10

```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```

a	b
10	20

```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```

a	b	soma
10	20	

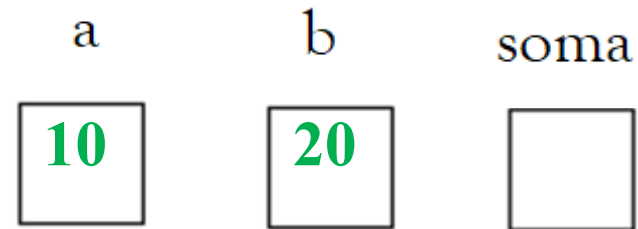
```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```





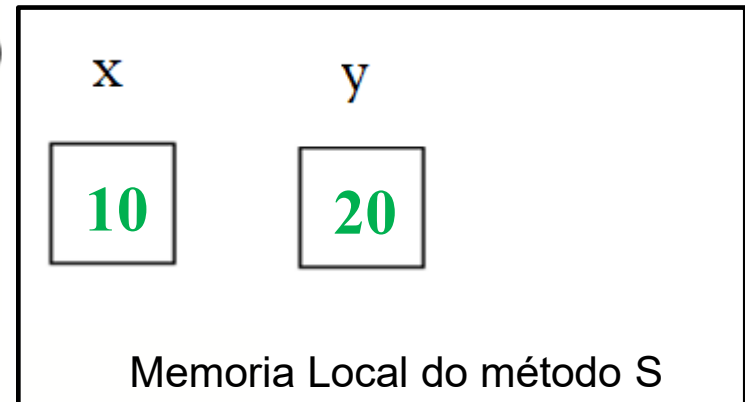
# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```



→ 

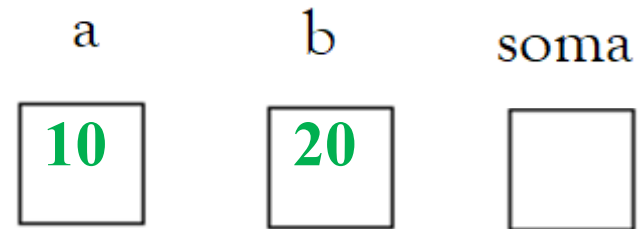
```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



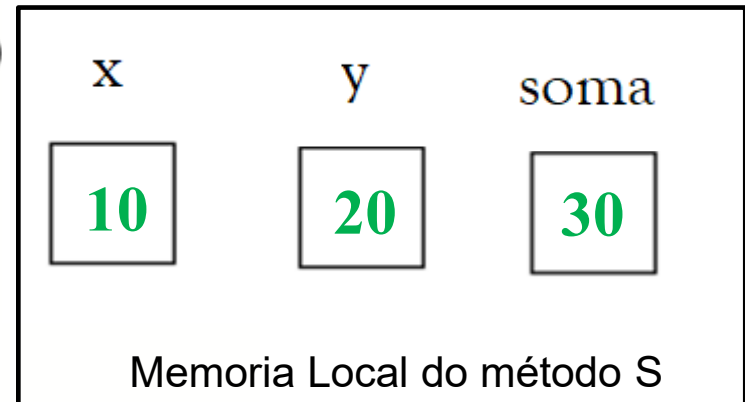


# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```



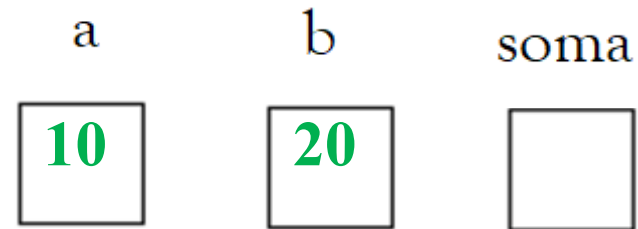
```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



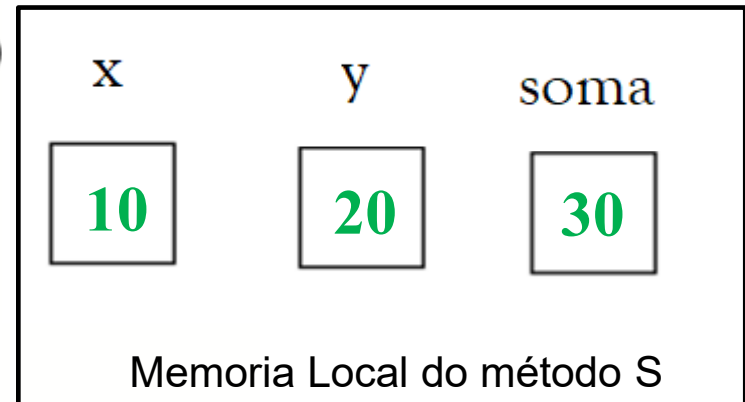


# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```



```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```





# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```

a	b	soma
10	20	30

```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



# Uso de Método com Retorno

```
public static void Main (string[] args){  
    int a = 10;  
    int b = 20;  
    int soma = S (a, b);  
    Console.WriteLine (soma);  
}
```

a	b	soma
10	20	30

```
public static int S (int x,int y)  
{  
    int soma = x + y;  
    return soma;  
}
```



# Uso de Método com Retorno

```
public static void Main (string[] args){
```

```
    int a = 10;
```

```
    int b = 20;
```

```
    int soma = S (a, b);
```

```
    Console.WriteLine (soma);
```

a

b

soma

10

20

30

```
}
```

```
public static int S (int x,int y)
```

```
{
```

```
    int soma = x + y;
```

```
    return soma;
```

```
}
```



# Métodos para Cálculos Matemáticos

- Dentre os vários recursos existentes na linguagem C#, pode-se destacar a classe Math, que possui uma série de funções para auxiliar diversas operações matemáticas.
- A classe Math possui propriedades e métodos, podendo-se destacar as propriedades E, PI e
- Métodos trigonométricos `Acos()`, `Asin()`, `Atan()`, `Cos()`, `Sin()` e `Tan()`;
- Métodos exponenciais `Pow()` e `Sqrt()`;
- Métodos logarítmicos `Exp()` e `Log()` e
- Possui também um conjunto com métodos para manipulação numérica, tais como `Abs()`, `Floor()`, `Max()`, `Min()` e `Round()`.



# Métodos para Cálculos Matemáticos

```
public static void Main (string[] args)
{
    Console.WriteLine(Math.Acos(-1));
    Console.WriteLine(Math.Asin(-1));
    Console.WriteLine(Math.Atan(1));
    Console.WriteLine(Math.Sin(1));
    Console.WriteLine(Math.Tan(4));
    Console.WriteLine(Math.Pow(2, 3));
    Console.WriteLine(Math.Sqrt(9.0));
    Console.WriteLine(Math.Exp(1));
    Console.WriteLine(Math.Log(3));
    Console.WriteLine(Math.Abs(-9));
    Console.WriteLine(Math.Floor(2.03));
    Console.WriteLine(Math.Max(10, 3));
    Console.WriteLine(Math.Min(10, 3));
    Console.WriteLine(Math.Round(5.8));
    Console.WriteLine(Math.PI);
    Console.WriteLine(Math.E);
}
```

```
3.14159265358979
-1.5707963267949
0.785398163397448
0.841470984807897
1.15782128234958
8
3
2.71828182845905
1.09861228866811
9
2
10
3
6
3.14159265358979
2.71828182845905
```





# Métodos para Cadeias de Caracteres

- As cadeias de caracteres são sequências alfanuméricas delimitadas como caracteres entre aspas dupla.
- Para o tratamento desse tipo de ação usam-se propriedades e métodos associados a um objeto (variável) do tipo string.
- Para uma demonstração desse recurso usam-se a propriedade Length e os métodos ToLower(), ToUpper() e Substring() que, respectivamente, apresentam o tamanho do string, o string escrito com caracteres minúsculos, o string escrito com caracteres maiúsculos e a extração de partes de um string.
- O método Substring() usa dois parâmetros, sendo o primeiro a posição do string a ser extraída e o segundo parâmetro define a quantidade de caracteres a ser extraída



# Métodos para Cadeias de Caracteres

```
public static void Main (string[] args)
{
    String TEXTO = "Computador";
    Console.WriteLine(TEXTO.Length);
    Console.WriteLine(TEXTO.ToLower());
    Console.WriteLine(TEXTO.ToUpper());
    Console.WriteLine(TEXTO.Substring(0, 3));
    Console.WriteLine(TEXTO.Substring(3, 4));
    Console.WriteLine(TEXTO.Substring(7, 3));
}
```

```
10
computador
COMPUTADOR
Com
puta
dor
```



# Exercícios

1. Fazer método que recebe 2 números double e retorna a multiplicação dos números.
2. Fazer método que calcule o valor da área de um retângulo, a partir do valor da base e altura.
3. Fazer método C# que recebe 1 número double e retorna a raiz quadrada do numero
4. Fazer método C# que recebe 1 vetor de número inteiros e retorna a quantidade de números impares
5. Fazer método C# que recebe 1 vetor de número inteiros e retorna o vetor invertido