

Introdução

Nesta semana iremos abordar um assunto muito interessante que poderá ajudar e muito no desenvolvimento organizado de um projeto.

Funções são as estruturas que permitem ao usuário separar seus programas em blocos.

Para conseguirmos desenvolver programas grandes e complexos, temos de construí-los bloco a bloco.

Em linguagem de programação estruturada o nome utilizado são funções e procedimentos.

Agora, em linguagem de programação orientada a objetos, que é o caso do nosso curso, chamaremos de métodos, pois é o nome dado quando usado este paradigma.

Os métodos são criados apenas uma vez e podem ser executados várias vezes em um programa, ou seja, podem ser reutilizados.

SINTAXE: Métodos

```
static tipo_de_retorno nome_do_método (declaração_de_parâmetros)
{
    corpo do método
}
```

O tipo-de-retorno é o tipo de variável que a função vai retornar. Caso tenha algum valor para ser retornado, usa-se o comando **return**.

A declaração de parâmetros é uma lista com a seguinte forma geral:

tipo var1, tipo var2, ... , tipo varN

O tipo deve ser especificado para cada uma das **N** variáveis de entrada. É na declaração de parâmetros que informamos ao compilador quais serão as entradas do método.

O corpo do método é onde as entradas são processadas, saídas são geradas ou outras instruções são realizadas.

Comando return

O comando ***return*** possui a seguinte forma geral: ***return valor_de_retorno;***

Digamos que uma função está sendo executada. Quando se chega a uma instrução ***return***, o método é encerrado imediatamente e, se o valor de retorno é informado, a função retorna este valor.

É importante lembrar que o valor de retorno fornecido tem que ser compatível com o tipo de retorno declarado para a função.

Caso o tipo de retorno for ***void*** (vazio), não é usada a instrução ***return*** no corpo da função, caso contrário, o uso é obrigatório.

Exemplo 1 – Método sem retorno de valor

Neste exemplo, estaremos usando três métodos:

O ***main()*** já é conhecido por nós, pois é o método principal, por onde a execução do programa se inicia.

O método ***digite()*** que será responsável apenas por exibir uma mensagem na tela.

E o método ***dobro(int n)***, que será responsável por receber um número inteiro por parâmetro, calcular e exibir o resultado do dobro de um número qualquer digitado pelo usuário.

O que estes dois novos métodos têm em comum?

Ambos não retornam valor, ou seja, o tipo de retorno é ***void***.

Exemplo 1 – Método sem retorno de valor – Código fonte

```
static void Main(string[] args)
{
    int a;
    digite();
    a = int.Parse(Console.ReadLine());
    dobro(a);
    Console.WriteLine("\n\nPressione alguma tecla.");
    Console.ReadKey();
}

static void digite()
{
    Console.Write("Digite um numero: ");
}

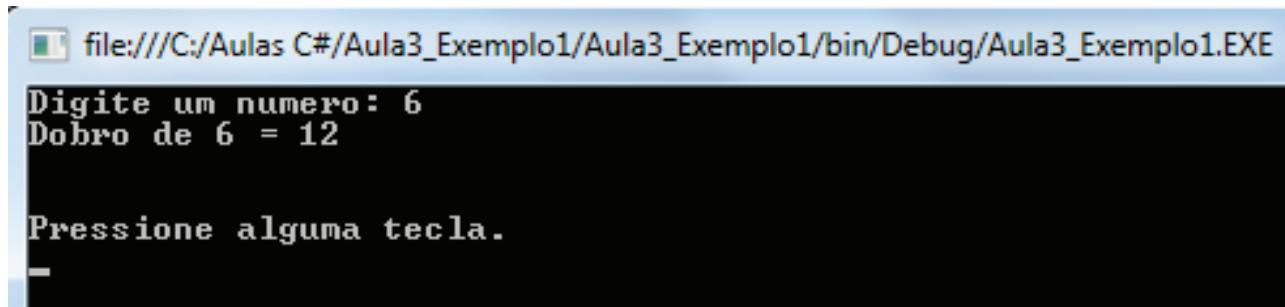
static void dobro(int n)
{
    int d = n * 2;
    Console.WriteLine("Dobro de {0} = {1}", n, d);
}
```

Método **main**

Método **digite**

Método **dobro**

Exemplo 1 – Método sem retorno de valor – Resultado



```
file:///C:/Aulas C#/Aula3_Exemplo1/Aula3_Exemplo1/bin/Debug/Aula3_Exemplo1.EXE
Digite um numero: 6
Dobro de 6 = 12

Pressione alguma tecla.
```

O usuário digitou um número inteiro e o programa exibiu o resultado do cálculo do dobro deste número.

Estes são apenas exemplos simples para mostrar como são realizadas as criações e chamadas para a execução de métodos. Veremos que a utilização de métodos é fundamental para organização e redução de linhas de código.

Exemplo 1 – Método sem retorno de valor – Analisando

Quando este programa é executado, são executadas as seguintes instruções na sequencia:

1. Declaração da variável **a** do tipo **int**.
2. É realizada a chamada do método **digite()**. Quando é feito isso, serão executadas as instruções que foram desenvolvidas dentro deste método, ou seja, será exibida a mensagem “*Digite um número:*” e na sequencia, continua a executar as instruções que estão dentro do método, retornará para o **main()**.
3. Na sequência o usuário digitará um número que será armazenado na variável **a**.
4. O próximo passo é realizada a chamada do método **dobro(a)**, ou seja, como o método **dobro** foi criado para receber um número inteiro, calcular e exibir o resultado do calculo, podemos dizer que esta chamada significa “calcular e exibir o dobro do valor que está na variável **a**”.
5. Sendo assim serão executadas as instruções que estão dentro do método **dobro**, que é calcular e exibir o resultado.
6. Finalizando as instruções do método **dobro**, é dada sequencia da execução das instruções do método **main**.

Fazendo uma analogia

Observem que foram criados sub-rotinas especializadas em realizar instruções específicas e com isso o método principal, ficou com menos responsabilidades, ou seja, ele apenas realizou uma chamada para a execução de um método, mas ele não precisa saber quais são as sequencias de comandos que este método vai executar.

Fazendo uma analogia: Você trabalha em uma empresa e seu chefe manda você ir comprar um chocolate. Você como é funcionário obediente vai comprar o chocolate e retorna entregando o chocolate para seu chefe.

Analizando este cenário, o chefe seria o **main** e você seria o método **comprar**. Interessou para o chefe saber onde você foi comprar o chocolate? O que importa para o chefe é que a tarefa foi executada com sucesso. E como ele sabe que você vai resolver isso para ele, o mesmo não precisa se preocupar como você realizou a tarefa, se foi no bar, padaria, mercado, ou ainda se pagou em dinheiro, cartão, cheque, ou se foi de carro, ônibus ou trem. Simplesmente ele mandou executar.

Passagem de parâmetros por valor

A linha de cabeçalho do método dobro é: ***static void dobro(int n)***

Quando foi realizada a chamada do método dobro, foi colocado entre parênteses a variável ***a***. Pois é o dobro do valor desta variável que o método irá calcular.

Observem que no cabeçalho desta função existe uma variável declarada. Esta variável ***n*** é quem receberá uma cópia do valor da variável ***a*** e assim o método será executado.

Caso aconteça algum problema com a variável ***n***, como por exemplo ela mudar de valor por conta de algum erro de programação, não afetará a variável ***a***, mantendo o valor original desta variável.

Exemplo 2 – Método com retorno de valor

Neste exemplo, estaremos usando três métodos:

O ***main()*** já é conhecido por nós, pois é o método principal, por onde a execução do programa se inicia.

O método ***digite()*** que será responsável apenas por exibir uma mensagem na tela.

E o método ***int tamanho(string x)***, que será responsável por receber uma string por parâmetro, calcular e retornar a quantidade de caracteres que possui esta string digitada pelo usuário.

O que estes dois novos métodos têm de diferentes?

Um não retornará valor (***void***) e o outro retornará um valor inteiro (***int***).

Exemplo 2 – Método com retorno de valor – Código fonte

```
static void Main(string[] args)
{
    int t;
    string p;
    digite();
    p = Console.ReadLine();
    t = tamanho(p);
    Console.WriteLine("\n{0} possui {1} caracteres", p, t);
    Console.WriteLine("\n\nPressione alguma tecla.");
    Console.ReadKey();
}

static void digite()
{
    Console.Write("Digite uma palavra: ");
}

static int tamanho(string x)
{
    return x.Length;
}
```

Exemplo 2 – Método com retorno de valor – Resultado

```
file:///C:/Aulas C#/Aula3_Exemplo2/Aula3_Exemplo2/bin/Debug/Aula3_Exemplo2.EXE
Digite uma palavra: inconstitucionalissimamente
inconstitucionalissimamente possui 27 caracteres
Pressione alguma tecla.
```

Neste exemplo foi digitada uma palavra e na sequencia foi exibida a quantidade de caracteres que possui esta palavra.

Poderia ser implementado sem usar o método tamanho? Sim, poderia. Mas lembrando que está apenas sendo demonstrado como é realizado o trabalho com métodos.

Exemplo 2 – Método com retorno de valor – Analisando

Quando este programa é executado, são executadas as seguintes instruções na sequencia:

1. Declaração das variáveis **p** e **t**.
2. É realizada a chamada do método **digite()**. Quando é feito isso, serão executadas as instruções que foram desenvolvidas dentro deste método, ou seja, será exibida a mensagem “*Digite uma palavra:*” e na sequencia, continua a executar as instruções que estão dentro do método, retornará para o **main()**.
3. Na sequência o usuário digitará uma palavra que será armazenado na variável **p**.
4. O próximo passo é realizada a chamada do método **tamanho(p)**, ou seja, como o método **tamanho** foi criado para receber uma string, verificar e retornar o resultado do calculo, podemos dizer que esta chamada significa “verificar e retornar a quantidade de caracteres da string que está na variável **p**.
5. Sendo assim serão executadas as instruções que estão dentro do método **tamanho**.
6. Finalizando as instruções do método **tamanho**, é dada sequencia da execução das instruções do método **main**, exibindo a mensagem para o usuário.

Métodos - Esclarecimentos

Agora com alguns exemplos já mostrados, ficou mais claro o objetivo da criação de métodos.

Com eles podemos organizar melhor o código fonte, ele pode ser chamado a qualquer momento e várias vezes durante a execução de um programa, mas é óbvio que vai depender da lógica de programação para o desenvolvimento.

Algumas instruções que utilizamos até agora, vão ficar mais claras para entender o que não são métodos, pois o tempo todo estamos utilizando-os, como por exemplo:

Console.**Write()**, Console.**Clear()**, Console.**ReadLine()** e Console.**.ReadKey()**

Observem que o que está em destaque, são todos métodos já pré-definidos pela classe Console, e não importa para nós como é que o **Clear()** faz para limpar a tela, o importante é que ele faz, ou seja, não precisamos nos preocupar quanto as instruções utilizadas por ele para realizar tal ação. E com isso agiliza o desenvolvimento.

Exemplo 3 – Método com tipos de Parâmetros Diferentes

Neste exemplo, o usuário terá que digitar o nome e a idade de duas pessoas. Um método chamado **PessoaMaisVelha** receberá estas informações e retornará uma string informando o nome da pessoa mais velha.

```
static void Main(string[] args)
{
    int idade1, idade2;
    string nome1, nome2;

    Console.Write("Digite o nome da 1º pessoa: ");    nome1 = Console.ReadLine();
    Console.Write("\nDigite a idade da 1º pessoa: ");  idade1 = int.Parse(Console.ReadLine());

    Console.Write("\n\nDigite o nome da 2º pessoa: "); nome2 = Console.ReadLine();
    Console.Write("\nDigite a idade da 2º pessoa: ");  idade2 = int.Parse(Console.ReadLine());

    string texto = PessoaMaisVelha(nome1, idade1, nome2, idade2);

    Console.WriteLine("\n" + texto);
    Console.WriteLine("\n\nPressione alguma tecla.");
    Console.ReadKey();
}

static string PessoaMaisVelha(string n1, int id1, string n2, int id2)
{
    if (id1 > id2)
        return n1 + " é a pessoa mais velha.";
    else
        if (id2 > id1)
            return n2 + " é a pessoa mais velha.";
        else
            return n1 + " e " + n2 + " tem a mesma idade.";
}
```

Exemplo 3-Método com tipos de Parâmetros Diferentes-Resultado

Sempre quando forem executar um programa, onde existem várias possibilidades de resposta, testem todas elas para verificar se estão coerentes, pois uma resposta incorreta ou incompleta, pode causar vários Transtornos, seja para um usuário, uma empresa ou até mesmo um outro sistema.

```
file:///C:/Aulas C#/Aula3_Exemplo3/Aula3_Exemplo3/
Digite o nome da 1º pessoa: Edilson
Digite a idade da 1º pessoa: 45

Digite o nome da 2º pessoa: Nivia
Digite a idade da 2º pessoa: 33
Edilson é a pessoa mais velha.

Pressione alguma tecla.
```

```
file:///C:/Aulas C#/Aula3_Exemplo3/Aula3_Exemplo3/
Digite o nome da 1º pessoa: Marcelo
Digite a idade da 1º pessoa: 22

Digite o nome da 2º pessoa: Elaine
Digite a idade da 2º pessoa: 27
Elaine é a pessoa mais velha.

Pressione alguma tecla.
-
```

```
file:///C:/Aulas C#/Aula3_Exemplo3/Aula3_Exemplo3/
Digite o nome da 1º pessoa: Rafael
Digite a idade da 1º pessoa: 16

Digite o nome da 2º pessoa: Telma
Digite a idade da 2º pessoa: 16
Rafael e Telma tem a mesma idade.

Pressione alguma tecla.
```

Exemplo 3 – Tipos de Parâmetros Diferentes – Comentários

As informações que foram digitadas, foram armazenadas nas variáveis **nome1** e **idade1** para referenciar a 1^a pessoa e nas variáveis **nome2** e **idade2** para referenciar a 2^a pessoa.

Logo após as entradas de dados, o método **PessoaMaisVelha** é invocado e são passadas por parâmetros cópia dos valores das variáveis **nome1**, **idade1**, **nome2**, **idade2** para as variáveis declaradas no cabeçalho do método que são **n1**, **id1**, **n2**, **id2**, respectivamente.

Este método irá realizar testes para saber quem é a pessoa mais velha e retornará uma string informando o ocorrido, sendo ela armazenada na variável **texto**. No enunciado não consta a informação que deve ser tratado o caso das idades forem iguais, porém isso deve ser pensado e tratado mesmo assim por quem está resolvendo este problema.

Estes detalhes devem sim serem interpretados e analisados para resolução de problemas, pois se não houver tratamento, o usuário poderá receber uma resposta errada do programa.

Exemplo 4 – Teste de Condição com retorno de método

Neste exemplo, o usuário terá que digitar um número inteiro e o programa informará se este número digitado pelo usuário é par ou ímpar.

```
static void Main(string[] args)
{
    int num;
    Console.Write("Entre com numero: ");
    num = int.Parse(Console.ReadLine());

    if (RestoPorDois(num)==0)
        Console.WriteLine("\n\nO numero é par.\n");
    else
        Console.WriteLine("\n\nO numero é impar.\n");

    Console.ReadKey();
}

static int RestoPorDois(int a)
{
    return a % 2;
}
```

Exemplo 4 – Teste de Condição com retorno de método-Resultado

Logo após o usuário digitar o número, foi realizado um teste condicional.

Para este teste, o método ***RestoPorDois*** foi invocado passado num como parâmetro.

Este método retorna um valor inteiro (0 ou 1) que será comparado com o valor 0 (zero) da condição (***if***).

Se o valor que o método retornar for 0 (zero), significa que o número é par, caso contrário, o número é ímpar.

```
file:///C:/Aulas C#/Aula3_Exemplo4/
Entre com numero: 78
O numero é par.
```

```
file:///C:/Aulas C#/Aula3_Exemplo4/
Entre com numero: 97
O numero é ímpar.
```

Bibliografia

- Manzano, José Augusto N. G., **Estudo Dirigido de Microsoft Visual C# 2010 Express.**
São Paulo, SP, Editora Érica, 2010.
- MSDN, Microsoft. **Guia de Programação C#.** Disponível:
[http://msdn.microsoft.com/pt-br/library/67ef8sbd\(v=vs.80\).aspx](http://msdn.microsoft.com/pt-br/library/67ef8sbd(v=vs.80).aspx). Acesso em 31 jan 2013
- <http://pt.wikipedia.org/wiki/Indenta%C3%A7%C3%A3o>