Curso Técnico

de Programação de Jogos Digitais

Aula 05

índice

[1. Métodos e Funções em C# 3](#_Toc169073784)

[2. Utilizando métodos anônimos 4](#_Toc169073785)

[3. Uso de função 5](#_Toc169073786)

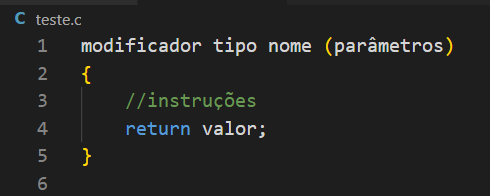
[4. Referencias 7](#_Toc169073787)

# Métodos e Funções em C#

Um método é um bloco de código que executa uma tarefa específica, podendo receber parâmetros de entrada e retornar um valor de saída. Um método é definido dentro de uma classe ou de uma estrutura, e pode ser chamado por meio de um objeto ou de uma referência da classe ou da estrutura.

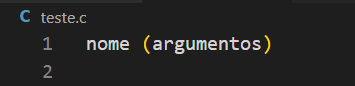
Uma função é um tipo especial de método que retorna um valor. Uma função pode ser usada como uma expressão, podendo ser atribuída a uma variável, passada como parâmetro ou usada em uma operação. Uma função pode ser definida como um método normal, como um método anônimo, como uma expressão lambda ou como um delegado.

Para declarar um método em C#, é preciso seguir a seguinte sintaxe:



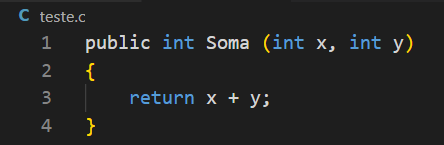
Onde modificador é a palavra-chave que define o nível de acesso do método (public, private, protected, internal, etc.), tipo é o tipo de dado do valor de retorno do método (void se o método não retornar nada), nome é o identificador do método, parâmetros são os valores de entrada do método, separados por vírgula, instruções são as linhas de código que executam a tarefa do método e valor é o valor de retorno do método.

Para chamar um método em C#, é preciso usar o nome do método seguido pelos argumentos entre parênteses, usando a seguinte sintaxe:



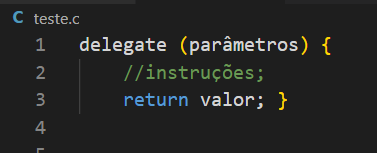
Onde nome é o identificador do método e argumentos são os valores que serão passados para os parâmetros do método, separados por vírgula. Os argumentos devem ter o mesmo tipo e a mesma ordem dos parâmetros.

Para declarar uma função como um método normal em C#, é preciso seguir a mesma sintaxe de um método, mas com um tipo diferente de void para o valor de retorno. Por exemplo:



Essa função recebe dois inteiros como parâmetros e retorna a soma deles.

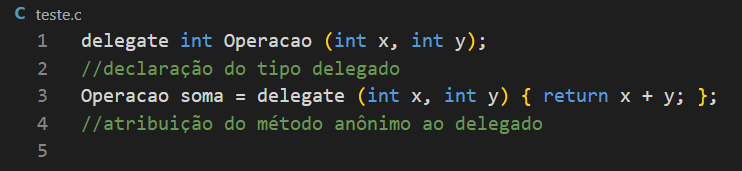
Para declarar uma função como um método anônimo em C#, é preciso usar a palavra-chave delegate seguida pelos parâmetros entre parênteses e pelo corpo da função entre chaves, usando a seguinte sintaxe:



Onde parâmetros são os valores de entrada da função, separados por vírgula, instruções são as linhas de código que executam a tarefa da função e valor é o valor de retorno da função. O tipo de retorno da função é inferido pelo compilador.

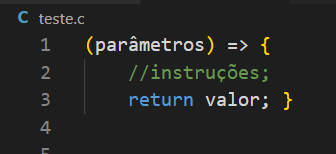
# Utilizando métodos anônimos

Um método anônimo pode ser atribuído a uma variável do tipo delegado, que é um tipo especial que representa uma referência a um método. Por exemplo:



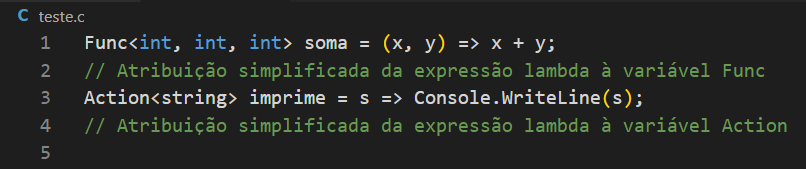
Essa declaração cria uma variável soma do tipo Operacao, que recebe uma referência ao método anônimo que retorna a soma de dois inteiros.

Para declarar uma função como uma expressão lambda em C#, é preciso usar os parâmetros seguidos pelo operador => e pelo corpo da função, usando a seguinte sintaxe:



Onde parâmetros são os valores de entrada da função, separados por vírgula, instruções são as linhas de código que executam a tarefa da função e valor é o valor de retorno da função. O tipo de retorno da função é inferido pelo compilador.

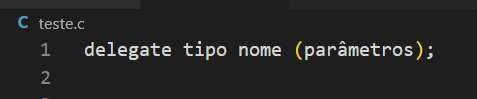
Uma expressão lambda pode ser atribuída a uma variável do tipo delegado ou a uma variável do tipo Func ou Action, que são tipos genéricos que representam funções com ou sem retorno. Por exemplo:



Essa declaração cria duas variáveis: soma, que recebe uma referência à expressão lambda que retorna a soma de dois inteiros, e imprime, que recebe uma referência à expressão lambda que imprime uma string no console.

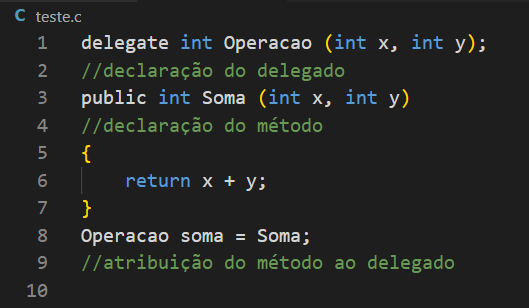
# Uso de função

Para declarar uma função como um delegado em C#, é preciso usar a palavra-chave delegate seguida pelo tipo de retorno, pelo nome da função e pelos parâmetros entre parênteses, usando a seguinte sintaxe:



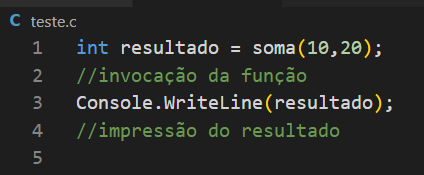
Onde tipo é o tipo de dado do valor de retorno da função, nome é o identificador da função e parâmetros são os valores de entrada da função, separados por vírgula.

Um delegado pode ser usado para criar uma referência a um método que tenha a mesma assinatura (tipo de retorno e parâmetros) do delegado. Por exemplo:



Essa declaração cria uma variável soma do tipo Operacao, que recebe uma referência ao método Soma que retorna a soma de dois inteiros.

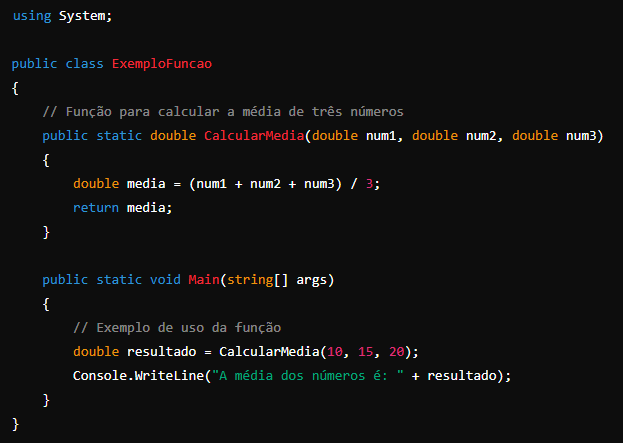
Para invocar uma função em C#, é preciso usar o nome da variável que contém a referência à função seguido pelos argumentos entre parênteses, usando a mesma sintaxe de um método. Por exemplo:



Essa invocação chama a função soma com os argumentos 10 e 20, e armazena o valor de retorno na variável resultado. Em seguida, imprime o valor de resultado no console.

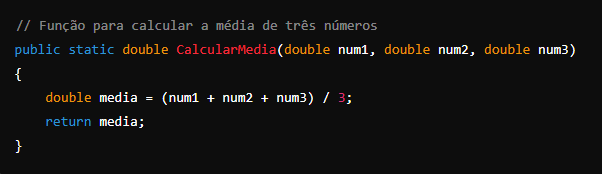
# Exemplo de Função

Vejamos um exemplo simples em C que usa uma função para calcular a média de três números. Este exemplo será dividido em partes para explicar cada parte do código.



Explicação do Código:

Definição da Função `CalcularMedia`



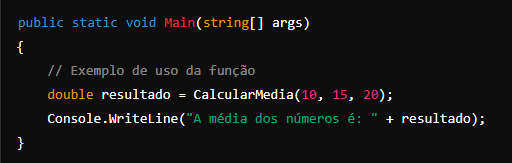
A função `CalcularMedia` é pública (`public`), o que significa que pode ser acessada de qualquer parte do código. É estática (`static`), o que permite chamar a função sem precisar criar uma instância da classe. Retorna um valor do tipo `double`, que é o tipo da média calculada.

**Parâmetros**: A função recebe três parâmetros do tipo `double` (`num1`, `num2` e `num3`). Esses parâmetros são os números para os quais queremos calcular a média.

**Cálculo da Média**: Dentro da função, calculamos a média dos três números usando a fórmula `(num1 + num2 + num3) / 3`. O resultado é armazenado na variável `media`.

**Função `return media;`:** A função retorna o valor calculado da média.

Função `Main` (Ponto de Entrada do Programa)



`public static void Main(string[] args)`: Este é o método `Main`, que é o ponto de entrada do programa. É aqui que o programa começa a ser executado.

`double resultado = CalcularMedia(10, 15, 20);`: Nesta linha, chamamos a função `CalcularMedia` com os argumentos `10`, `15` e `20`. Isso significa que estamos calculando a média dos números `10`, `15` e `20` e armazenando o resultado na variável `resultado`, que é do tipo `double`.

`Console.WriteLine("A média dos números é: " + resultado);`: Por fim, usamos `Console.WriteLine` para imprimir o resultado na tela. Aqui, concatenamos a string `"A média dos números é: "` com o valor de `resultado` convertido para string.

# Referencias

1. Unity Technologies. (2020). **Unity User Manual.** Unity Technologies.
2. [C#, 2020] **Visual C# Developer Center**, Microsoft Docs.

Microsoft Corporation. (2020).

1. Geig, M. (2018). **Unity 2018 Game Development in 24 Hours**, Sams Teach Yourself. Sams Publishing.
2. Hocking, J. (2015). **Unity in Action: Multiplatform Game Development in C#.** Manning Publications.