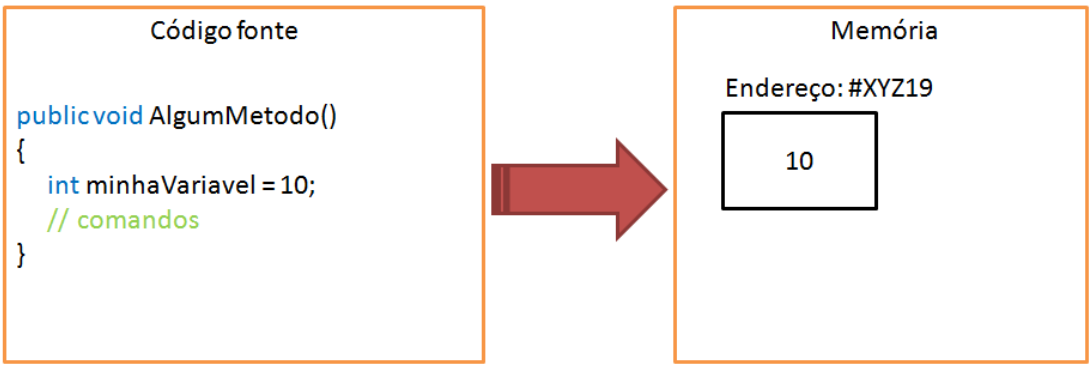
O que são Delegates (C#)?

Vamos aprender alguns recursos do C# que aparentemente são taxados como “muito complexos”. Neste momento, vamos falar sobre delegates. Para não ficar muito complexo, o assunto foi dividido em 3 partes. Ao longo das aulas vamos aprender mais sobre esses recursos interessantíssimos e melhorar ainda mais nosso trabalho no dia a dia. Então, vamos ao estudo!

Vamos aprender um pouco sobre os famosos **Delegates**. Os delegates, basicamente, são tipos da linguagem c# que permitem que façamos uma referência para métodos. Parece complexo, mas vamos entender o que significa isso de uma forma mais simples. Observem a ilustração abaixo:



Declaração de Variável

No exemplo, podemos ver a simplicidade de uma declaração de variável e o que acontece na memória. Declaramos uma variável do tipo “int” e então atribuímos o valor 10. Para esta variável é criado um endereço de memória que conterá o valor 10 e este endereço de memória aponta para a variável de nome “minhaVariável”.

Muito interessante, sei como funciona a declaração de variáveis, qual a relação com os delegates? Calma, chegaremos lá.

Uma variável inteira aceita inteiros, um delegate aceita métodos. Isso quer dizer que podemos guardar em nossos delegates métodos. E com isso podemos executá-los no momento que desejarmos, em tempo de execução. Vamos verificar a sintaxe da declaração de um delegate:

```
<Modificador de acesso> <Tipo de Retorno> delegate <Nome do Delegate>([Parametros]);
```

A declaração de um delegate, assemelha-se muito a uma assinatura de um método, com exceção da palavra-chave “delegate”. Vamos ver maiores detalhes sobre a declaração do delegate:

- Modificador de acesso:
    - public, protected, private – mas é interessante que seja public
- Tipo de Retorno:
    - Qualquer tipo de retorno que desejamos para obter algum resultado com a necessidade. Pode ser algum tipo específico como classes, structs, tipos primitivos, etc.

- delegate:
    - palavra-chave necessária para se declarar um delegate;
- Nome do delegate:
    - Nome que irá representar o seu delegate. Eu costume nomeá-lo com um sufixo "Handler". Exemplo. ClickHandler.
- Parâmetros:
    - Opcional, mas caso necessário, parâmetros podem ser adicionados à assinatura do seu delegate.
    - Ao declarar um delegate, temos um novo tipo de "dados" que possibilita guardar referências de métodos com a mesma assinatura.

Vejamos um exemplo:

```
public void delegate AposExecutarCalculo();
```

Neste exemplo criamos um delegate que não retorna nenhum valor e tem uma assinatura que não necessita de parâmetros. Neste momento, criamos um novo "tipo" que poderá conter métodos que não retornam valores e não necessitam de parâmetros. Vamos declarar uma variável do tipo delegate criado:

```
public AposExecutarCalculo meuDelegate;
```

Se parece muito com uma declaração simples de uma variável inteira, não parece? Esta variável aceita qualquer tipo de valor? Não. Somente métodos que possuem a mesma assinatura do delegate. Agora, vamos fazer um exemplo de atribuição de métodos para o nosso delegate:

1 – criar um método:

```
public void ImprimirInformacao()
{
        Console.WriteLine("Terminou a execução do processo");
}
```

2 – Associar ao delegate:

```
meuDelegate += new AposExecutarCalculo(ImprimirInformacao);
```

Pronto. Simples não? Agora o nosso delegate aponta para um método, no caso ImprimirInformacao. Com isso é possível executar o método a qualquer momento. Podemos observar também o operador "+=". Ele é utilizado para associar 1 ou mais métodos ao meu delegate. Isso é maravilhoso porque com isso podemos fazer com que apenas 1 chamada ao nosso delegate execute todos os métodos associados a ele. Existe também a possibilidade de utilização do operador "-=" que nos permite desvincular algum método do delegate. Como executar o método ImprimirInformacao a partir do meu delegate?

Simples:

```
meuDelegate(); //Executando o método
```

Olha, que interessante, utilizei o meu delegate como se fosse um método. Legal de mais né? Isso acontece porque o delegate aponta para o endereço de memória que está localizado nosso método ImprimirInformacao. Como o método ImprimirInformacao possui a mesma assinatura do nosso delegate, então, efetuar a chamada da nossa "variável especial" como um método é como se estivéssemos chamando o ImprimirInformacao, só que com um apelido.

Se observarmos, notamos que a forma de declaração de uma "variável" do tipo delegate assemelha-se e muito à declaração de uma variável do tipo inteiro, conforme nosso exemplo. Também o fato de que os métodos atribuídos ao delegate devem possuir a mesma assinatura é bem parecido com o exemplo do tipo inteiro.