

# UNIDADE 2 - Tema 1 - Construtores de Classe



Microfundamento: Algoritmos e abstração de dados

Eixo (<https://pucminas.instructure.com/courses/49042>) > Microfundamento

(<https://pucminas.instructure.com/courses/49011/pages/microfundamento-algoritmos-e-abstracao-de-dados>) > UNIDADE 2 - Tema 1 - Construtores de Classe

## CONSTRUTORES DE CLASSE

Construtores de Classe são métodos especiais que temos para serem executados no momento da criação de um objeto. Fornece-nos muita flexibilidade e é fundamental sabermos definir, e muito bem, um construtor e, inclusive, sobrecarregá-lo.

### NESTA SESSÃO

#### Construtores e Destrutores de Classes

Construtores são métodos especiais responsáveis pela criação de um objeto e pela inicialização de valores para todos os seus atributos. Um construtor é executado toda vez que um objeto é instanciado (palavra-chave *new*).

Se um Construtor não for definido pelo programador, o C# (CLR) se encarrega de criar um, por padrão. O objetivo é criar um objeto e colocá-lo em estado válido para ser utilizado.

As variáveis dentro de uma classe, por padrão, são inicializadas com os valores:

- 0 (zero) para todos os tipos numéricos, *int*, *double* etc.;
- false para tipos booleanos;
- '0' (*null*) para tipos *string*.

Construtores são métodos que possuem o mesmo nome da classe na qual são declarados.

Eles não possuem tipo de retorno, são normalmente declarados como públicos (*public*), podem receber parâmetros, assim como qualquer método, e podem ser sobrecarregados. Veja um exemplo:

```
class Círculo
{
    private double _Raio;

    public double Raio
    {
        get { return _Raio; }
        set { _Raio = value; }
    }

    public Círculo()
    {
        _Raio = 0;
    }

    public double CalculaÁrea()
    {
        return Math.PI * Math.Pow(_Raio, 2);
    }
}
```

Nesse momento, se fizermos

```
Círculo xCírculo = new Círculo();
```

o Construtor será chamado e atribuirá ao campo `_Raio` o valor 0.

Construtores podem ser sobrecarregados. Em outras palavras, podemos ter mais de um Construtor, desde que eles tenham números de parâmetros diferentes, ou tipos de parâmetros diferentes, ou ordem de parâmetros diferentes. Poderíamos fazer isso, por exemplo:

```
class Círculo
{
    private double _Raio;

    public double Raio
    {
        get { return _Raio; }
        set { _Raio = value; }
    }

    public Círculo()
    {
        _Raio = 0;
    }

    public Círculo(int R)
    {
        _Raio = R;
    }

    public double CalculaÁrea()
    {
        return Math.PI * Math.Pow(_Raio, 2);
    }
}
```

Você pode observar que definimos dois Construtores, um sem parâmetro e outro com um

parâmetro R inteiro. Então se fizermos:

```
Círculo xCírculo = new Círculo();
```

inicializaremos o campo `_Raio` com 0. Mas se fizermos:

```
Círculo xCírculo = new Círculo(3);
```

inicializaremos o campo `_Raio` com 3.

## Flexibilidade e controle

Destrutores são utilizados quando existe a necessidade de se codificar algum comportamento especial no momento em que um determinado objeto for destruído. Eles não são chamados ou referenciados, são invocados automaticamente.

Normalmente o programador não define um Destrutor. O CLR se encarrega de fornecer um padrão que orientará as ações de eliminação de um dado objeto.

Uma classe só pode ter um destrutor. São métodos que possuem também o mesmo nome da classe em que são declarados, mas com o caractere `~` (til) precedendo esse nome. Eles não possuem tipo de retorno e não possuem parâmetros:

```
~Círculo()  
{  
    // Algum Código...  
}
```

Não existe um comando explícito em C# para eliminar um objeto. O Destrutor é chamado pelo *Garbage Colector*, ou Coletor de Lixo.

Assista ao vídeo abaixo:





Muito bem! Você compreendeu bem esse conteúdo. A partir de agora temos condições de elaborar classes muito sofisticadas e usá-las em nossos programas.

Isso termina a nossa sessão de estudos e agora é só praticar.

### REFLEXÃO

Em outros exemplos realizados em outras sessões, como você poderia melhorá-los utilizando construtores?




APÓS CONCLUIR ESTA SESSÃO DE ESTUDO, RESPONDA:

**Desenvolva uma classe que, por meio de métodos, realize as seguintes operações dentro de um vetor de números inteiros:**

- Criação de um vetor de dimensões fornecidas pelo usuário segundo um método construtor. Se não for fornecido pelo usuário o tamanho deverá ser, por padrão, 10;
- Criação de um vetor de dimensões fornecidas pelo usuário e inserção automática de valores aleatórios nesse vetor, segundo um método construtor, fornecidos os limites mínimo e máximo; Se não fornecido o tamanho do vetor pelo usuário o tamanho deverá ser 10;
- Listagem de todos os elementos do vetor;
- Inserção de um valor em uma dada posição do vetor;
- Recuperação de um valor indicado por uma posição fornecida pelo usuário;
- Localização do Maior e do Menor número dentro do vetor.

**Teste as rotinas no programa principal (*main*).**

Clique [aqui](https://pucminas.instructure.com/courses/49011/files/2899718?wrap=1) (<https://pucminas.instructure.com/courses/49011/files/2899718?wrap=1>)  ([https://pucminas.instructure.com/courses/49011/files/2899718/download?download\\_frd=1](https://pucminas.instructure.com/courses/49011/files/2899718/download?download_frd=1)) para conferir o gabarito.

[↑ Voltar ao topo](#)

## TIPOS ABSTRATOS DE DADOS - CLASSES - IMPLEMENTAÇÃO

### Tema 1: Tipos Abstratos de Dados

#### **Definição de um TAD - Classes e Objetos**

<https://pucminas.instructure.com/courses/49011/pages/unidade-2-tema-1-definicao-de-um-tad-classes-e-objetos>

#### **Atributos, Propriedades e Métodos de Classe**

<https://pucminas.instructure.com/courses/49011/pages/unidade-2-tema-1-atributos-propriedades-e-metodos-de-classe>

#### **Mecanismos de Visibilidade/Acessibilidade**

<https://pucminas.instructure.com/courses/49011/pages/unidade-2-tema-1-mecanismos-de-visibilidade-slash-acessibilidade>

#### **Construtores de Classe**

**Síntese e referências**

(<https://pucminas.instructure.com/courses/49011/pages/unidade-2-sintese-e-referencias>)

**Atividade objetiva 2**

(<https://pucminas.instructure.com/courses/49011/quizzes/171709>)