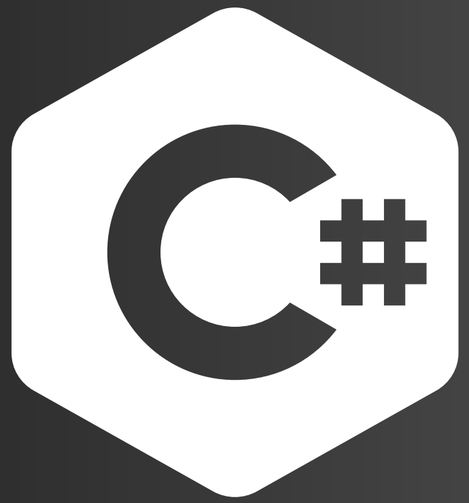


Variáveis, Tipos de Dados e Escopo

Prof. Wanderson Timóteo
www.wandersontimoteo.com.br

O que vamos aprender:

- O que são variáveis?;
- Conceitos Fundamentais de Variáveis em C#;
- Tipos de Dados em C#:
  - Primitivos;
  - Não Primitivos.
- Método Console.WriteLine;
- Escopo de Variáveis em C#;
- Comentários;

O que vamos aprender:

- **O que são variáveis?;**
- ~~Conceitos Fundamentais de Variáveis em C#;~~
- ~~Tipos de Dados em C#:~~
  - ~~Primitivos;~~
  - ~~Não Primitivos.~~
- ~~Método Console.WriteLine;~~
- ~~Escopo de Variáveis em C#;~~
- ~~Comentários;~~

Wanderson Timóteo

O que são Variáveis?

Na programação, uma variável é um espaço reservado na memória do computador utilizado para armazenar dados que podem variar durante a execução de um programa. Esses dados podem ser números, textos, valores lógicos (verdadeiro ou falso) ou outros tipos de informação.

Para facilitar a manipulação desses dados, as variáveis são associadas a nomes, chamados de **identificadores**, que são escolhidos pelo programador. ***Esses identificadores são usados para referenciar os valores armazenados na memória***.

Uma vez declarada e inicializada com um valor, uma variável pode ser usada em diversas operações dentro do programa, como cálculos, comparações e atribuições. As variáveis são fundamentais para a programação, pois permitem que os programas sejam dinâmicos e possam lidar com diferentes tipos de dados.

O que vamos aprender:

- ~~O que são variáveis?;~~
- **Conceitos Fundamentais de Variáveis em C#;**
- ~~Tipos de Dados em C#:~~
  - ~~Primitivos;~~
  - ~~Não Primitivos.~~
- ~~Método Console.WriteLine;~~
- ~~Escopo de Variáveis em C#;~~
- ~~Comentários;~~

Wanderson Timóteo

Wanderson Timóteo

Conceitos Fundamentais de Variáveis em C#

Declaração de Variáveis:

Para declarar uma variável em C#, precisamos definir o **tipo de dado** que ela vai armazenar e o **identificador** (*nome da variável*). A sintaxe básica para declarar uma variável é:

```
tipoDeDado nomeVariavel;
```

tipoDeDado: define o tipo de dados que a variável pode armazenar, por exemplo, números, texto entre outros.
nomeVariavel: é o nome dado à variável, que será usado para referenciá-la no código.

Conceitos Fundamentais de Variáveis em C#

Atribuição de Valor a Variável:

Para dar um valor inicial a uma variável, usamos o operador de atribuição **= (igual)**.
A sintaxe é:

```
tipoDeDado nomeVariavel;
nomeVariavel = valor;
```

Quando usar a atribuição?

A atribuição é comum quando o valor que será armazenado na variável não é conhecido no momento da sua declaração e precisa ser definido posteriormente no código.

Conceitos Fundamentais de Variáveis em C#

Inicialização de Variável

A inicialização de uma variável é o ato de dar um valor inicial a ela no momento da sua declaração. Quando você inicializa uma variável, você está criando-a e atribuindo a ela um valor ao mesmo tempo. Isso garante que a variável comece a ser utilizada com um valor definido. Sintaxe:

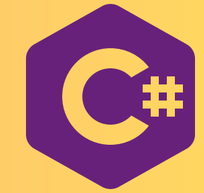
```
tipoDeDado nomeVariavel = valor;
```

Por que inicializar?

Inicializar variáveis é uma boa prática, pois garante que elas não terão valores indefinidos ou não atribuídos antes de serem usadas, o que poderia resultar em erros no código.

O que vamos aprender:

- ~~O que são variáveis?;~~
- ~~Conceitos Fundamentais de Variáveis em C#;~~
- **Tipos de Dados em C#:**
  - **Primitivos;**
  - **Não Primitivos.**
- ~~Método Console.WriteLine;~~
- ~~Escopo de Variáveis em C#;~~
- ~~Comentários;~~



Wanderson Timóteo

Tipos de Dados em C#

Os tipos de dados em C# são classificados em:

- **Primitivos,**
- **Não Primitivos**.

Eles definem **que tipo de valor** uma variável pode armazenar e como esse valor é representado na memória. Eles determinam o tamanho e a forma dos dados que podem ser armazenados. C# é uma linguagem **fortemente tipada**, o que significa que ***cada variável precisa ter um tipo definido no momento da declaração***.

Tipos de Dados em C#

Tipos Primitivos

São tipos de dados básicos que são fornecidos diretamente pela linguagem e armazenam valores simples. Eles têm suporte direto pelo compilador e ocupam um espaço fixo na memória.

1. **Tipos Numéricos Inteiros;**
2. **Tipos Numéricos de Ponto Flutuante;**
3. **Tipo Caracter;**
4. **Tipos Lógicos;**
5. **Outros...**

Vamos analisar alguns.

Tipos de Dados Primitivos em C#

1. Tipos Numéricos Inteiros:

Short: Armazena números inteiros menores de até 16 bits.

```
short ano = 2025;
```

int (inteiro): Armazena números inteiros, positivos ou negativos de até 32 bits, geralmente de -2.147.483.648 a 2.147.483.647.

```
int contador = 10;
```

long: Armazena números inteiros maiores de até 64 bits, aproximadamente -9 quatrilhões a 9 quatrilhões..

```
long populacaoMundial= 7800000000;
```

Wanderson Timóteo

Tipos de Dados Primitivos em C#

2. Tipos Numéricos de Ponto Flutuante:

float: Armazena números decimais de menor precisão até 32 bits. Requer o sufixo **f** ao atribuir um valor.

```
float temperatura = 43.6f;
```

double: Armazena números decimais de maior precisão até 64 bits.

```
double precoProduto = 199.90;
```

Wanderson Timóteo

Tipos de Dados Primitivos em C#

3. Tipo Caracter:

char: Armazena um único caractere, ou seja, armazena apenas uma letra, um número ou um símbolo.

```
char inicial = "W";
```

Tipos de Dados Primitivos em C#

4. Tipos Lógicos:

bool: Armazena valores lógicos, sendo true (verdadeiro) ou false (falso), representando lógica booleana.

True

```
bool estaLogado = true;
```

False

```
bool ativo = false;
```

Wanderson Timóteo

Tipos de Dados em C#

Tipos Não Primitivos

São mais complexos e podem ser construídos a partir dos tipos primitivos. Eles geralmente armazenam referências para um objeto, em vez do próprio valor.

1. **Strings;**
2. **Arrays;**
3. **Classes;**
4. **Structs;**
5. **Enums;**
6. **Interfaces;**
7. **Delegates;**
8. **Nullable Types;**
9. **Object.**

Vamos analisar alguns.

Tipos de Dados Não Primitivos em C#

1. Strings:

String: Representa uma sequência de caracteres. Ao contrário de char, que armazena um único caractere, string é usada para armazenar textos maiores.

```
string saudacao = "Olá mundo”;
```

2. Arrays:

Arrays: Um array é uma coleção de elementos do mesmo tipo armazenados em posições contíguas de memória. Pode ser de qualquer tipo primitivo ou não primitivo.

```
int[] numeros = { 1, 2, 3, 4, 5 };
string[] nomes = { "Wanderson", "Ryan", "Jeane" };
```

Wanderson Timóteo

Tipos de Dados Não Primitivos em C#

3. Classes

Classes: Definem um tipo que pode ter campos, propriedades, métodos e eventos. Classes são a base da programação orientada a objetos em C#.

```
public class Pessoa
{
    public string nome;
    public int idade;
}
```

Tipos de Dados Não Primitivos em C#

4. Structs

Structs: Parecidas com classes, mas geralmente são usadas para criar tipos que contêm pequenos grupos de dados. Ao contrário das classes, structs são tipos por valor.

```
public class Ponto
{
    public int X;
    public int Y;
}
```

Wanderson Timóteo

Tipos de Dados Não Primitivos em C#

5. Enums

Enums: São usados para definir conjuntos de constantes nomeadas, o que torna o código mais legível quando se lida com um conjunto de valores fixos.

```
public enum DiaDaSemana
{
    Segunda, Terça, Quarta, Quinta, Sexta, Sábado, Domingo
}
```

Tipos de Dados Não Primitivos em C#

6. Interfaces

Interfaces: Especificam um contrato que classes ou structs podem implementar, definindo métodos ou propriedades que devem ser implementados pela classe.

```
public interface  IVeiculo
{
    void Mover();
}
```

Wanderson Timóteo

Tipos de Dados Não Primitivos em C#

7. Delegates

Delegates: Representam referências a métodos, permitindo que métodos sejam passados como parâmetros e chamados indiretamente. Eles são essenciais para eventos e programação orientada a eventos.

```
public delegate void Processar(int numero);
```

Wanderson Timóteo

Tipos de Dados Não Primitivos em C#

8. Nullable Types

Nullable Types: Permitem que tipos por valor como int, double, etc., possam ser nulos (null), o que normalmente é possível apenas para tipos por referência.

```
int? idade = null;
```

idade pode ser um valor ou null

Tipos de Dados Não Primitivos em C#

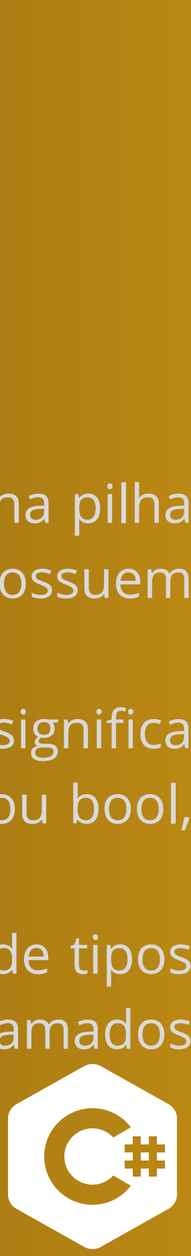
Esses tipos **não primitivos** são amplamente usados em aplicações C# para criar estruturas de dados complexas, manipular listas, gerenciar comportamentos de objetos, e lidar com cenários onde os dados podem ser opcionais ou desconhecidos. Eles são fundamentais para a construção de aplicações robustas e escaláveis em .NET.

Prof. Wanderson Timóteo E-mail: wanderson.timoteo@mt.senac.br

Tipos de Dados em C#

Diferença entre Tipos Primitivos e Não Primitivos

- **Armazenamento**: Tipos primitivos geralmente são armazenados diretamente na pilha (stack), enquanto os tipos não primitivos são armazenados na heap e possuem referências na pilha.
- **Imutabilidade**: Alguns tipos não primitivos como string são imutáveis, o que significa que, uma vez criados, não podem ser modificados. Tipos primitivos, como int ou bool, são valores simples e podem ser alterados diretamente.
- **Referência vs. Valor**: Tipos primitivos em C# são frequentemente chamados de tipos por valor, já que armazenam diretamente os dados. Tipos não primitivos são chamados de tipos por referência, pois armazenam uma referência para um local na memória onde o valor está armazenado.



Wanderson Timóteo

O que vamos aprender:

- ~~O que são variáveis?;~~
- ~~Conceitos Fundamentais de Variáveis em C#;~~
- ~~Tipos de Dados em C#;~~
  - ~~Primitivos;~~
  - ~~Não Primitivos.~~
- **Método Console.WriteLine;**
- ~~Escopo de Variáveis em C#;~~
- ~~Comentários;~~

Wanderson Timóteo

Método Console.WriteLine

O **Console.WriteLine** é um método em C# que pertence à classe Console, e é usado para exibir mensagens no console (ou terminal) de uma aplicação. Esse método é uma forma simples e direta de imprimir texto ou valores de variáveis na tela durante a execução de programas de console, sendo muito útil para debugging, interação simples com o usuário e exibição de resultados.

Método Console.WriteLine

Estrutura Básica do Console.WriteLine:

```
Console.WriteLine("Mensagem a ser exibida");
```

Console: A classe Console faz parte do namespace System, que contém funcionalidades básicas do sistema em C#. Ela fornece métodos para entrada e saída de dados no console.

WriteLine: O método WriteLine é responsável por imprimir uma mensagem no console e mover o cursor para a próxima linha após exibir o texto. Isso significa que, depois de usar WriteLine, qualquer texto subsequente aparecerá em uma nova linha.

Wanderson Timóteo

Wanderson Timóteo

O que vamos aprender:

- ~~O que são variáveis?;~~
- ~~Conceitos Fundamentais de Variáveis em C#;~~
- ~~Tipos de Dados em C#;~~
  - ~~Primitivos;~~
  - ~~Não Primitivos.~~
- ~~Método Console.WriteLine;~~
- **Escopo de Variáveis em C#;**
- ~~Comentários;~~

Tipos de Escopo de Variáveis em C#

O escopo de variáveis em C# define a região do código onde uma variável é visível e pode ser acessada. Ele determina onde a variável pode ser usada, onde pode ser modificada e quando deixa de existir. O escopo é importante para garantir que variáveis não sejam acidentalmente alteradas em lugares onde não deveriam, ajudando a prevenir erros.

Existem diferentes tipos de escopo em C#:

1. **Escopo Local (Variáveis Locais),**
2. **Escopo de Bloco (Controle de Fluxo),**
3. **Escopo de Parâmetros de Método,**
4. **Escopo de Campos de Classe (Variáveis de Instância),**
5. **Escopo de Variáveis Estáticas.**

Tipos de Escopo de Variáveis em C#

1. Escopo Local (Variáveis Locais):

Variáveis locais são aquelas que são declaradas dentro de um método, bloco de código ou função. Elas são visíveis e podem ser acessadas apenas dentro do bloco onde foram definidas. Um bloco é delimitado por **chaves {}**.

Tempo de Vida:

A variável local existe apenas enquanto o bloco de código em que foi declarada está sendo executado. Quando o bloco termina, a variável é destruída e a memória é liberada.

```
void MeuMetodo()
{
    int x = 10;
    Console.WriteLine(x);
}
```

No exemplo, 'x' é uma variável local e só existe dentro de 'MeuMetodo', ou seja, ‘x’ só é visível dentro do método e não pode ser acessado fora do método.

Wanderson Timóteo

Tipos de Escopo de Variáveis em C#

2. Escopo de Bloco (Controle de Fluxo):

Dentro de estruturas de controle de fluxo (como **if, for, while**, etc.), variáveis podem ser declaradas, e o escopo delas fica restrito ao bloco específico onde foram criadas.

Tempo de Vida: A variável existe apenas dentro do bloco de controle.

Exemplo:

```
for (int i = 0; i < 5; i++)
{
    int y = i * 2;
    Console.WriteLine(y);
}
// Console.WriteLine(i, y); Não são acessíveis fora do bloco
```

Neste caso, tanto **i** quanto **y** têm escopo limitado ao bloco do **for**, ou seja, eles só existem dentro do bloco **for** e não estão acessíveis fora dele.

Prof. Wanderson Timóteo E-mail: wanderson.timoteo@mt.senac.br

Tipos de Escopo de Variáveis em C#

3. Escopo de Parâmetros de Método:

Parâmetros são variáveis que são passadas para um método quando ele é chamado. Esses parâmetros têm escopo dentro do próprio método.

Tempo de Vida: Os parâmetros são destruídos quando a execução do método termina.
Exemplo:

No exemplo, ***mensagem*** só existe dentro do método **ExibirMensagem** e não pode ser usada fora dele.

```
void ExibirMensagem(string mensagem)
{
    Console.WriteLine(mensagem);
}
ExibirMensagem("Olá mundo!");
```

Wanderson Timóteo

Tipos de Escopo de Variáveis em C#

4. Escopo de Campos de Classe (Variáveis de Instância):

Campos são variáveis que são declaradas diretamente dentro de uma classe ou estrutura e fora de qualquer método. Eles são chamados de variáveis de instância ou de campo e podem ser acessados por qualquer método da classe.

Tempo de Vida: O tempo de vida de um campo é o mesmo da instância da classe que o possui. Ele existe enquanto o objeto está em memória.

Tipos de Campos:

- **Campos de Instância:** Pertencem a uma instância específica de uma classe.
- **Campos Estáticos:** Pertencem à classe em si, não a uma instância específica,

e são compartilhados por todas as instâncias.

Wanderson Timóteo

Tipos de Escopo de Variáveis em C#

4. Escopo de Campos de Classe (Variáveis de Instância):

Exemplo de Campos de Instância:

```
class Pessoa
{
    public string Nome;
}
Pessoa pessoa1 = new Pessoa();
pessoa1.Nome = "Wanderson";
```

Nome: Variável de instância.

pessoa1.Nome: 'Nome' pertence à instância 'pessoa1' da classe 'Pessoa'.

Tipos de Escopo de Variáveis em C#

4. Escopo de Campos de Classe (Variáveis de Instância):

Exemplo de Campos Estáticos:

```
class Configuracao
{
    public  static string  Ambiente = "Desenvolvimento";
}
Console.WriteLine(Configuracao.Ambiente);
```

Ambiente: Variável estática.

Acessa sem precisar de uma instância.

Tipos de Escopo de Variáveis em C#

5. Escopo de Variáveis Estáticas:

Variáveis estáticas são aquelas que pertencem à classe em si, e não a uma instância específica. Isso significa que o valor delas é compartilhado por todas as instâncias da classe.

Tempo de Vida: Variáveis estáticas têm tempo de vida que coincide com o tempo de execução do aplicativo, ou seja, existem durante toda a execução do programa. Nesse exemplo, TotalInstancias é um campo estático que conta quantas instâncias da classe Contador foram criadas.

```
class Contador
{
    public static int TotalInstancias = 0;
    public Contador()
    {
        TotalInstancias++;
    }
}
Contador c1 = new Contador();
Contador c2 = new Contador();
Console.WriteLine(Contador.TotalInstancias);
```

Saída: 2, pois a variável é compartilhada.

Wanderson Timóteo

Resumo do Escopo de Variáveis

Tipo de Variável	Escopo	Tempo de Vida
Variáveis Locais	Dentro de métodos, funções ou blocos de código	Durante a execução do bloco ou método
Parâmetros de Método	Dentro do método que recebe os parâmetros	Durante a execução do método
Campos de Classe	Dentro de uma classe, fora dos métodos	Enquanto a instância da classe existir ou durante a execução da aplicação, no caso de estáticos.
Variáveis Estáticas	Acessível em qualquer lugar que tenha acesso à classe	Durante toda a execução da aplicação

Wanderson Timóteo

O que vamos aprender:

- ~~O que são variáveis?;~~
- ~~Conceitos Fundamentais de Variáveis em C#;~~
- ~~Tipos de Dados em C#;~~
  - ~~Primitivos;~~
  - ~~Não Primitivos.~~
- ~~Método Console.WriteLine;~~
- ~~Escopo de Variáveis em C#;~~
- **Comentários**

Comentários

Os comentários são usados para explicar o código, documentar funcionalidades ou anotações importantes. Eles são ignorados pelo compilador e não afetam a execução do programa.

Comentários

Comentários de Linha Única

Para comentar uma única linha, use //. Tudo o que estiver após // será considerado um comentário.

```
// Este é um comentário de linha única
string nome = "Wanderson"; // Comentário ao lado do código
```

Wanderson Timóteo

Comentários

Comentários de Múltiplas Linhas

Para comentar várias linhas, use /* ... */. Esse tipo de comentário é útil para trechos maiores de texto ou para "desativar" blocos de código.

```
/* Este é um comentário de múltiplas linhas que pode
abranger mais de uma linha */
string nome = "Wanderson"; // Comentário ao lado do código
```

Wanderson Timóteo

Fim!!!