
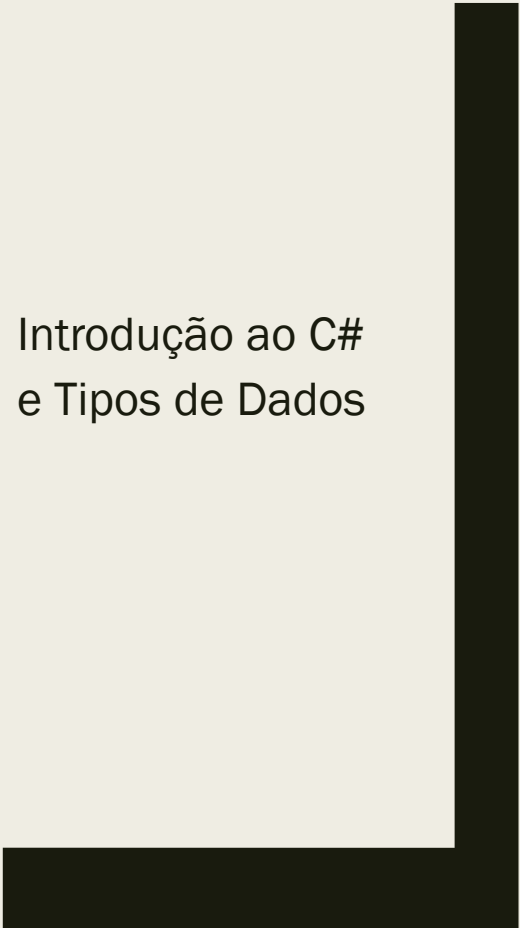




# FUNDAMENTOS DE C# – AULA 1



Introdução ao C#  
e Tipos de Dados



# Sobre mim

## Nathan Ferreira

- 8º Período Ciências da Computação
- 1 ano e meio de experiencia profissional em C# .NET
- Instagram, X, LinkedIn, Gmail:  
**nathanf10994**
- Whatsapp:  
**(31) 993-512-934**



# O que é C#?

- Criado pela Microsoft em 2000
- Linguagem orientada a objetos, moderna e fortemente tipada
- Multi-paradigma (OO, funcional, assíncrono)
- Usada em: Desktop, Web (ASP.NET), Mobile (MAUI), Jogos (Unity)



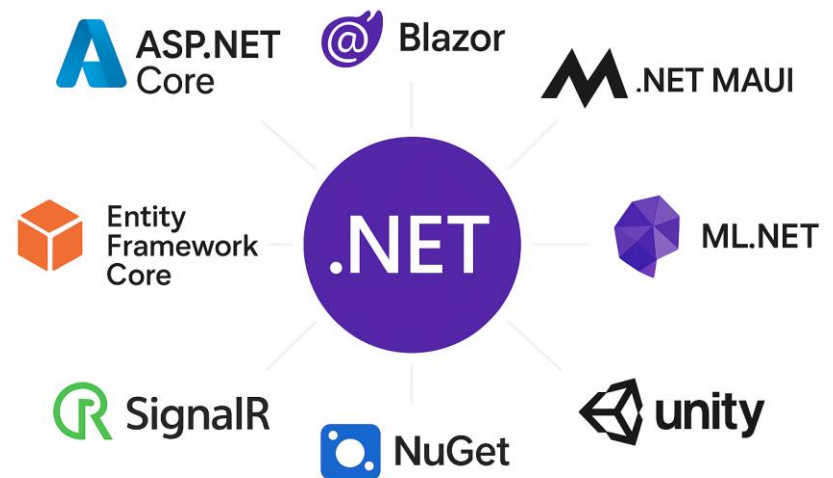
# O que é .NET?

- Plataforma de desenvolvimento multiplataforma e open source
- Suporte a várias linguagens: C#, F#, VB.NET
- Ferramentas: Visual Studio, VS Code, dotnet CLI
- Grande comunidade e bibliotecas disponíveis via NuGet



# Frameworks e tecnologias .NET

- **ASP.NET** - Aplicações Web, APIs REST e serviços backend.
- **Blazor** - Interfaces web interativas usando C# em vez de JavaScript.
- **MAUI (sucessor do Xamarin)** - Aplicações Mobile (iOS, Android) e Desktop (Windows, Mac).
- **WinForms e WPF** - Aplicações Desktop Windows tradicionais.
- **Unity** - Plataforma de desenvolvimento de jogos 2D/3D.
- **Entity Framework Core (EF Core)** - ORM (Object-Relational Mapper) para acesso a bancos de dados com LINQ.
- **NuGet** - Gerenciador de pacotes oficial do .NET com milhares de bibliotecas reutilizáveis.



# Estrutura de um Programa

```
using System; // Importa bibliotecas prontas do .NET

namespace HelloWorld // Organiza o código em um agrupamento lógico
{
    0 referências
    public class Program // Classe principal do projeto
    {
        0 referências
        public static void Main(string[] args) // Método principal, ponto de entrada do programa
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

# Tipos de Dados

- **Inteiros:** int, long, short, byte
- **Decimais:** float, double, decimal
- **Texto:** char, string
- **Booleano:** bool (true/false)
- **Nullable:** int? idade = null
  
- **C# é uma linguagem fortemente tipada** → o tipo da variável deve ser definido na declaração e não pode ser alterado depois.

# Variáveis e Constantes

```
int idade = 31;  
string nome = "Nathan";  
const double PI = 3.1415;  
bool ativo = true;  
  
var dado = new DateTime();
```

- **var** → inferência de tipo: O tipo da variável é definido pelo valor atribuído, por isso é obrigatório inicializar na declaração.
- **const** → valor fixo: uma vez atribuído um valor, ele não pode ser alterado.
- **readonly** → definido apenas no construtor



# Regras – Variáveis e Constantes

- Não podem começar com um número:

```
int 1numero; //errado
```

```
int numero1; //correto
```

- Não podem conter espaços:

```
int mes de nascimento; //errado
```

```
int mesDeNascimento; //correto
```

- Não podem usar caracteres especiais(exceto \_):

```
string $email; //errado
```

```
string _email; //correto
```

- Não podem usar palavras reservadas da linguagem:

```
string class; //errado
```

```
string classe; //correto
```

- Sensível a maiúsculas e minúsculas(case sensitive):

```
float numero; float Numero; //variáveis diferentes
```

# Boas práticas – Variáveis e Constantes

- Use nomes descritivos e claros: evite termos genéricos ou abreviações desnecessárias.
- Mantenha consistência: use o mesmo padrão de nomenclatura em todo o projeto.
- Inicialize variáveis sempre que possível no momento da declaração.
- Padrões de nomenclatura:
  - Variáveis:* **camelCase** → numeroDeUsuarios, nomeCompleto.
  - Constantes:* letras maiúsculas ou **PascalCase** → TAXA\_DE\_JUROS, MaximoTentativas.
- Defina o escopo mínimo necessário: declare variáveis o mais próximo possível do uso.
- Use **const** ou **readonly** para valores que não mudam, garantindo segurança no código.

# Boas práticas – Variáveis e Constantes

```
// Bons exemplos

int numeroDeUsuarios;

string nomeCompleto;

float precoDoProduto;

DateTime horarioDeEntrada;

List<string> clientesRegistrados;
```

```
// maus exemplos

int num1;

float x;

double v1;

string end;

List<int> lista1;
```

- Nomes curtos demais ou abreviados podem confundir. Prefira clareza e contexto, mesmo que o nome fique um pouco maior.

# PascalCase e camelCase

## PascalCase

- Cada palavra começa com letra maiúscula (inclusive a primeira).
- Não há separadores (\_ ou -), tudo junto.
- Exemplo:  
NomeCompleto, DataNascimento, NumeroTelefone
- Padrões de utilização em C#:  
Classes, Métodos, Propriedades, Constantes

## camelCase

- Primeira palavra começa com letra minúscula, as demais com maiúscula.
- Também sem separadores (\_ ou -).
- Exemplo:  
nomeCompleto, dataNascimento, numeroTelefone
- Padrões de utilização em C#:  
Variáveis locais, Parâmetros de métodos, Campos privados (às vezes com \_ no início)

# Conversões

- **Conversões Implícitas** - Feitas automaticamente pelo compilador, sem risco de perda de dados. Exemplo: de tipo menor para tipo maior (int → double).

```
int numeroInteiro = 10;  
double numeroFlutuante = numeroInteiro; // conversão implícita
```

- **Conversões Explícitas (Casting)** - Precisam ser feitas manualmente porque podem perder dados. Usa o operador (tipo).

```
double valorDecimal = 9.7;  
int numeroInteiro = (int)valorDecimal; // perde a parte decimal → 9
```

- **Classe Convert** - Conversão segura entre tipos (string, numéricos, boolean, etc.).

```
string texto = "123";  
int numero = Convert.ToInt32(texto);
```

```
double preco = 10.5;  
string precoTexto = Convert.ToString(preco);
```

- **Principais métodos:** ToInt32() → converte para int; ToDouble() → converte para double; ToDecimal() → converte para decimal; ToString() → converte para string; ToBoolean() → converte para bool

# Conversões

- **Métodos Parse** - Usados para converter string → tipo numérico (ou outro). Se a string não for válida, gera o erro (FormatException).

```
string texto = "3,14";  
double numero = double.Parse(texto);
```

- **Métodos TryParse** - Parecido com Parse, mas não gera erro se a conversão falhar. Retorna true ou false.

```
string texto = "abc"; int numero;  
  
if (int.TryParse(texto, out numero)) Console.WriteLine("Conversão OK: " + numero);  
else Console.WriteLine("Falha na conversão");
```

- **ToString()** - Converte qualquer valor para string. Pode formatar números, datas, moedas etc.

```
double valor = 1234.56;  
Console.WriteLine(valor.ToString("C")); // "R$ 1.234,56"
```

# Entrada e saída de dados

- **Console.WriteLine()** - Escreve uma saída no console e pula para a próxima linha. Aceita textos, variáveis e interpolação.

```
Console.WriteLine("Olá, Mundo!");  
Console.WriteLine($"A soma é {2 + 3}");
```

- **Console.Write()** - Igual ao WriteLine(), mas não pula linha após escrever.

```
Console.Write("Digite seu nome: ");
```

- **Console.ReadLine()** - Lê uma linha completa digitada pelo usuário e retorna uma string. Para atribuir a variáveis de outros tipos (int, double, etc.), é necessário fazer uma conversão.

```
string nome = Console.ReadLine(); // OK, já é string  
int idade = int.Parse(Console.ReadLine()); // conversão  
double altura = double.Parse(Console.ReadLine()); // conversão
```

# Entrada e saída de dados

- `Console.ReadKey()` - Lê uma tecla pressionada pelo usuário. Pode ser usado para pausar o programa até o usuário apertar algo.

```
Console.WriteLine("Pressione qualquer tecla para continuar...");  
Console.ReadKey();
```

- `Console.Clear()` → Limpa a tela do console.

```
Console.WriteLine("Olá, Mundo!");  
Console.ReadKey();  
Console.Clear(); // Limpa todo o conteúdo exibido anteriormente no console
```



# Exercícios

## ■ 1 – Dados Pessoais:

Receber como entrada do usuário: **nome**, **idade** e **cidade**.  
Exibir no console uma frase concatenando esses dados.

```
// Exercício 1

Console.Write("Informe seu nome completo -> ");
string nomeCompleto = Console.ReadLine();

Console.Write("Informe a cidade onde reside -> ");
string cidadeAtual = Console.ReadLine();

Console.Write("Informe sua idade -> ");
int idade = int.Parse(Console.ReadLine());

Console.WriteLine();
Console.Write($"Olá, meu nome é {nomeCompleto}. ");
Console.WriteLine($"Tenho {idade} anos. Moro em {cidadeAtual}.");
Console.WriteLine();
```

# Exercícios

## ■ 2 – Área do Círculo:

Receber do usuário o valor do **raio** de um círculo.

Utilizar a constante **PI (3.1415)** para calcular sua área, usando a fórmula:  $a = PI * (Raio * Raio)$ .

```
// exercício 2

const double PI = 3.1415;

Console.Write("informe o raio do círculo -> ");
double raioDoCirculo = double.Parse(Console.ReadLine());

double areaDoCirculo = PI * (raioDoCirculo * raioDoCirculo);

Console.WriteLine();
Console.WriteLine($"A área do círculo é {areaDoCirculo:F2}");
Console.WriteLine();
```

# Exercícios

## ■ 3 – Calculadora de IMC:

Receber do usuário o **peso** (em kg) e a **altura** (em metros).  
Calcular o **IMC** usando a fórmula:  $\text{peso} / (\text{altura} * \text{altura})$ .

```
// Exercício 3

Console.Write("Informe seu peso(em kilogramas) -> ");
double pesoEmKilogramas = double.Parse(Console.ReadLine());

Console.Write("Informe sua altura(em Metros) -> ");
double alturaEmMetros = double.Parse(Console.ReadLine());

double indiceDeMassaCorporal = pesoEmKilogramas / (alturaEmMetros * alturaEmMetros);

Console.WriteLine();
Console.WriteLine($"O seu IMC é {indiceDeMassaCorporal:F2}.");
Console.WriteLine();
```

# Exercícios

## ■ 4 - Operações Matemáticas:

Receber dois números do usuário e exibir na tela o resultado da soma, subtração, multiplicação e divisão entre eles.

```
//Exercício 4
Console.Write("Informe o primeiro número -> ");
double primeiroNumero = int.Parse(Console.ReadLine());

Console.Write("Informe o segundo número -> ");
double segundoNumero = int.Parse(Console.ReadLine());

Console.WriteLine();
Console.WriteLine($"A soma dos números informados " +
    $"resulta em {(primeiroNumero + segundoNumero)}");
Console.WriteLine($"A subtração dos números informados " +
    $"resulta em {primeiroNumero - segundoNumero}");
Console.WriteLine($"A multiplicação dos números informados " +
    $"resulta em {primeiroNumero * segundoNumero}");
Console.WriteLine($"A divisão dos números informados " +
    $"resulta em {primeiroNumero / segundoNumero}");
Console.WriteLine();
```

# Exercícios

## ■ 5. Tabuada de Multiplicação:

Solicite ao usuário um número inteiro.

Exiba a tabuada de multiplicação desse número de 1 a 10.

```
// Exercício 5
```

```
Console.Write("Informe o número para a tabuada de multiplicação-> ");  
int numeroParaTabuada = int.Parse(Console.ReadLine());
```

```
Console.WriteLine();
```

```
Console.WriteLine($"{numeroParaTabuada} x 01 = {numeroParaTabuada * 1}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 02 = {numeroParaTabuada * 2}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 03 = {numeroParaTabuada * 3}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 04 = {numeroParaTabuada * 4}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 05 = {numeroParaTabuada * 5}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 06 = {numeroParaTabuada * 6}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 07 = {numeroParaTabuada * 7}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 08 = {numeroParaTabuada * 8}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 09 = {numeroParaTabuada * 9}");
```

```
Console.WriteLine($"{numeroParaTabuada} x 10 = {numeroParaTabuada * 10}");
```

```
Console.WriteLine();
```

# Possíveis temas da próxima aula:

- Operadores Matemáticos
- Operadores de Atribuição
- Operadores de Comparação
- Operadores Lógicos
- Estruturas de Decisão (if-else e switch-case)
- Operadores Ternários