




Lógica de Programação

---



Tema | Lógica de programação

---

## Primeiro Dia.

15h00 - 15h05 | Abertura

15h05 - 15h15 | Welcome (Gestores Ada e Mercado Eletrônico)

15h15 - 15h25 | Quebra-gelo (Host Ada)

15h25 - 16h00 | Mesa Redonda (Dia-a-dia de um dev)

16h00 - 16h15 | Apresentação da Jornada + Orientações (Host Ada)

16h15 - 16h20 | Fechamento (Host Ada)

16h20 - 16h30 | Intervalo

16h20 - 18h00 | Início das atividades com instrutores

## C# (C Sharp)

O C# (c sharp) é uma linguagem de programação orientada a objetos criada pela Microsoft e tem como base as linguagens em C, o que traz uma similaridade entre as linguagens.

## C# (C Sharp)

O C# (c sharp) é uma linguagem de programação orientada a objetos criada pela Microsoft e tem como base as linguagens em C, o que traz uma similaridade entre as linguagens.

## Um pouco de História

Em 1999, no desenvolvimento da plataforma .NET, **Anders Hejlsberg** formou uma equipe de programadores para desenvolver uma nova linguagem de programação.

Sua sintaxe foi baseada no C++, mas inclui varias influências de outras linguagens de programação, como Object Pascal e Java.



## C# e o Java

C# - É a resposta da **Microsoft** ao Java. É todo OO e é linguagem base do .NET, tecnologia da **Microsoft** que bate de frente com o Java



O # de C# se refere ao sinal musical (sustenido), que aumenta em meio tom uma nota musical.

**Mas antes de tudo...**

## O que é um Framework?

Um ***framework*** (ou arcabouço), em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

**Em outras palavras:** É um monte de códigos que já estão prontos e você só precisa implementar.



## O DotNet

**.NET (ou DotNet)** é uma iniciativa da empresa Microsoft, que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para .NET pode ser executado em qualquer dispositivo que possua um framework de tal plataforma.

## SDK – Software Development Kit

O SDK (Software Development Kit) inclui tudo o que você precisa para criar e executar aplicativos .NET, usando ferramentas de linha de comando e qualquer editor (como Visual Studio)

## Cli do .Net

A CLI (interface de linha de comando) do .NET é uma ferramenta multiplataforma para desenvolvimento, criação, execução e publicação de aplicativos .NET.

A CLI do .NET está incluída no .NET SDK. Para saber como instalar o SDK do .NET, consulte [Instalar o .NET Core](#).

## Runtime do .NET Core

O termo "Runtime" do .NET Core refere-se ao ambiente de execução para aplicativos desenvolvidos na plataforma .NET Core. O .NET Core inclui uma versão específica do Common Language Runtime (CLR), que é responsável pela execução do código .NET Core em tempo de execução. Vamos explorar alguns pontos-chave relacionados ao "Runtime" do .NET Core:

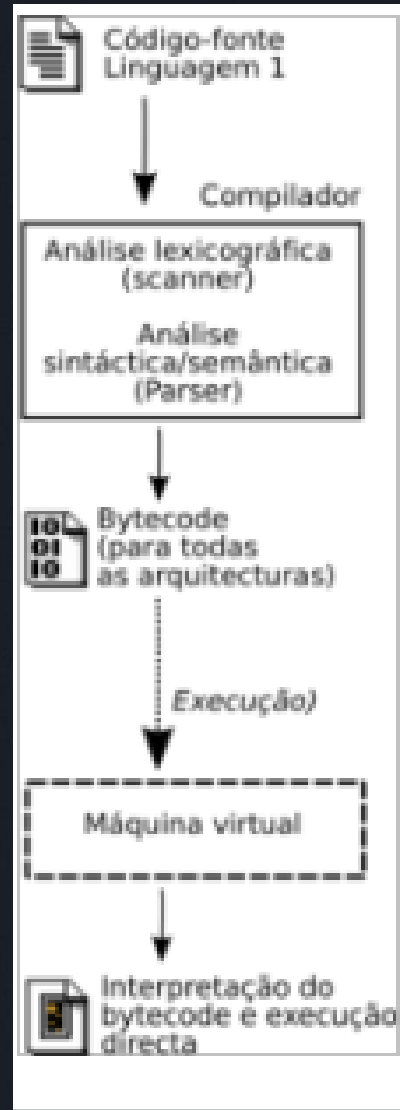
**Common Language Runtime (CLR):** A CLR no .NET Core é uma versão leve e modular do CLR que faz parte do framework .NET Core. Ela executa funções essenciais, como compilação Just-In-Time (JIT), gerenciamento de memória, coleta de lixo e execução de código intermediário (IL) em código de máquina nativo.

**Ambiente de Execução Multiplataforma:** Assim como o ASP.NET Core, o .NET Core é projetado para ser multiplataforma, o que significa que os aplicativos podem ser desenvolvidos e executados em diferentes sistemas operacionais, incluindo Windows, Linux e macOS. Isso é possível devido ao design modular e à portabilidade do .NET Core Runtime.

## SDK e Runtime do .Net

- O SDK é usado durante o desenvolvimento e inclui ferramentas para criar, compilar e testar aplicativos.
- O Runtime é necessário para executar aplicativos .NET Core e inclui a CLR e bibliotecas essenciais.
- O SDK é instalado no ambiente de desenvolvimento, enquanto o Runtime é instalado no ambiente de produção ou onde os aplicativos serão executados.

## Fluxo de Compilação e Interpretação do Código C#



## .Net Framework e .Net Core

### - Modelo de Desenvolvimento e Suporte a Plataformas:

**.NET Framework:** Tradicionalmente, o .NET Framework foi projetado principalmente para o desenvolvimento de aplicativos Windows e era específico para a plataforma Windows.

**.NET Core:** Foi desenvolvido com um foco maior na portabilidade e na execução em várias plataformas, incluindo Windows, Linux e macOS. O .NET Core foi projetado para ser modular e leve, permitindo a criação de aplicativos mais eficientes e portáteis.

### - Arquitetura e Componentes:

**.NET Framework:** Monolítico e integrado ao sistema operacional Windows. As versões específicas do .NET Framework precisam ser instaladas no sistema.

**.NET Core:** Mais modular, com a capacidade de incluir apenas os componentes necessários para um aplicativo específico. Ele pode ser implantado como uma parte do aplicativo, não interferindo no sistema operacional.

## .Net Framework e .Net Core

### - Suporte a Multiplataforma:

**.NET Framework:** Limitado ao Windows.

**.NET Core:** Projetado para ser multiplataforma, oferecendo suporte a Windows, Linux e macOS.

### - Versões e Evolução:

**.NET Framework:** Versões incrementais foram lançadas ao longo do tempo, mas não há uma versão única e unificada para todas as plataformas.

**.NET Core:** O .NET Core evoluiu para se tornar o .NET 5 e subsequentemente o .NET 6, unificando as diferentes variantes do .NET (incluindo o .NET Framework e o .NET Core) em uma única plataforma.



## Runtime do .NET Core

Compilação Just-In-Time (JIT): Durante a execução de um aplicativo .NET Core, o código IL gerado durante a compilação é traduzido para código de máquina nativo pela CLR no momento em que o aplicativo é iniciado. Isso é conhecido como compilação Just-In-Time (JIT) e permite otimizações específicas da máquina.

Módulos e Pacotes Leves: O .NET Core é modular, o que significa que você pode incluir apenas os componentes necessários para o seu aplicativo. Isso resulta em aplicações mais leves e eficientes, pois você pode escolher os módulos do .NET Core necessários para o seu aplicativo específico.

Independência do Sistema Operacional: O .NET Core é independente do sistema operacional, o que facilita a criação de aplicativos que podem ser executados em diferentes ambientes sem a necessidade de recompilar o código-fonte.

Em resumo, o "Runtime" do .NET Core consiste na CLR específica do .NET Core e em outros componentes que possibilitam a execução de aplicativos .NET Core em diferentes plataformas de maneira eficiente e portátil. Isso contribui para a flexibilidade e a capacidade de implantação em diversos ambientes.

# IDE - Integrated Development Environment

Em português: Ambiente de Desenvolvimento Integrado.

Para utilizar o C#, temos:

- [Visual Studio \(Microsoft\)](#)
- [Visual Studio Code \(Microsoft\)](#)
- [Rider \(JetBrains\)](#)

Visual Studio - Algumas categorias de Templates comuns incluem

### **ASP.NET Core**

Templates para aplicativos web usando ASP.NET Core.

### **Windows Forms e WPF**

Templates para o desenvolvimento de aplicativos de área de trabalho usando Windows Forms ou Windows Presentation Foundation (WPF).

### **Console App**

Templates para aplicativos de console em várias linguagens.

## Visual Studio - Algumas categorias de templates comuns incluem

### **Azure**

Templates para aplicativos e serviços em nuvem usando serviços Azure.

### **Mobile**

Templates para o desenvolvimento de aplicativos móveis, incluindo Xamarin.

### **Bibliotecas de Classes**

Templates para criar bibliotecas de classes reutilizáveis.

### **Test Projects**

Templates para projetos de teste, incluindo testes unitários e testes de interface do usuário.

### **Docker**

Templates para suporte a Docker, permitindo a criação de contêineres para aplicativos

# Organizando um sistema com C#

## Projeto

É uma unidade organizacional que agrupa todos os arquivos e recursos necessários para criar um aplicativo ou uma biblioteca

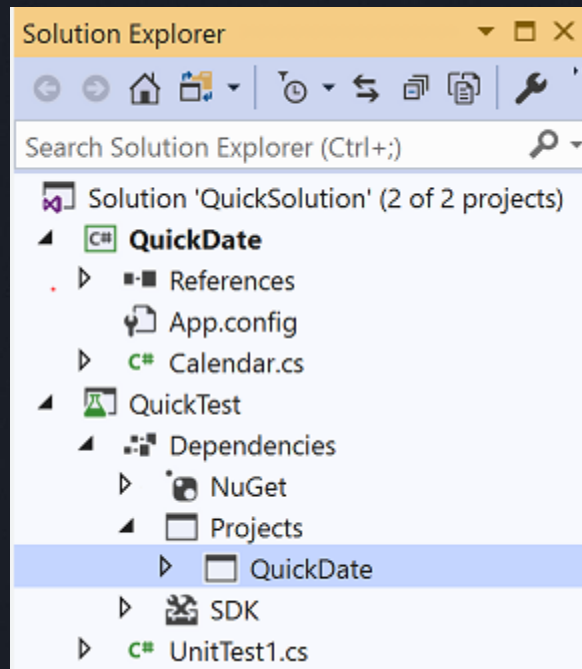
## Solution (Solução)

Contém um ou mais projetos relacionados.

## Namespace

Um namespace em C# é uma forma de organizar e controlar o escopo dos identificadores usados em um programa

## Organizando um sistema com C#



## Variáveis

Em programação, variável é um mecanismo que utilizamos para armazenar valor. Podemos dizer também que uma variável é um fragmento de memória no computador onde guardamos dados.

```
int valorInteiro = 50;  
int valorNegativoInteiro = -25;  
uint valorPositivoInteiro = 10;  
long valorGrandeInteiro = 1500;  
long valorNegativoGrande = -2450;  
ulong valorPositivoGrande = 3498;  
float valorQuebradoFloat = 1.47F;  
double valorQuebradoDouble = 7.98;  
double valorQuebradoDouble2 = 7.98D;  
decimal valorQuebradoDecimal = 9.32M;  
char valorChar = 'S';  
string valorTexto = "variável que guarda texto";
```

## Vetor ou Array

Um vetor, em inglês “array”, é um agrupamento de variáveis do mesmo tipo. Esse agrupamento possui um valor fixo de quantas posições podemos armazenar.

```
int[] vetorInteiro = new int[5];  
int[] vetorInteiro = new int[] { 1, 2, 3, 4, 5 };
```



## Vetor ou Array

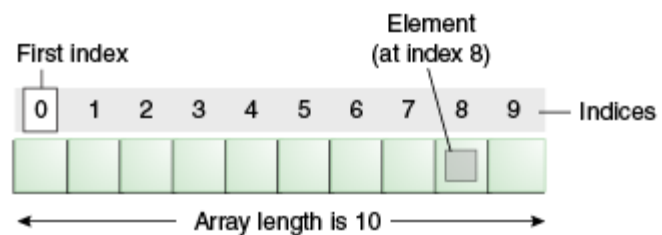
- Array de Arrays

```
int[][] jaggedArray = new int[6][];
```

```
int[][] jaggedArray =  
{  
    new int[] { 1, 2, 3 },  
    new int[] { 4, 5, 6, 7, 8 },  
    new int[] { 9, 10 },  
    new int[] { 11, 12, 13, 14 },  
    new int[] { 15 },  
    new int[] { 16, 17, 18, 19, 20, 21 }  
};
```

```
int[][] jaggedArray = new int[6][];  
  
// Atribuindo tamanhos às submatrizes  
jaggedArray[0] = new int[3]; // A primeira submatriz tem tamanho 3  
jaggedArray[1] = new int[5]; // A segunda submatriz tem tamanho 5  
jaggedArray[2] = new int[2]; // A terceira submatriz tem tamanho 2  
jaggedArray[3] = new int[4]; // A quarta submatriz tem tamanho 4  
jaggedArray[4] = new int[1]; // A quinta submatriz tem tamanho 1  
jaggedArray[5] = new int[6]; // A sexta submatriz tem tamanho 6  
  
// Acessando elementos da matriz irregular  
jaggedArray[0][0] = 10;  
jaggedArray[1][2] = 20;  
jaggedArray[3][1] = 30;
```

## Posição em um array ou índice



An array of 10 elements.

## Matriz

Uma matriz é uma coleção de elementos que são organizados em uma grade bidimensional.

A matriz é um vetor de posições horizontais e verticais

```
int[,] matrizInteiro = new int[4,5]
```

```
int[,] matrizInteiro = new int[,] {  
    { 1, 2, 3, 4, 5 },  
    { 6, 7, 8, 9, 10 },  
    { 11, 12, 13, 14, 15 },  
    { 16, 17, 18, 19, 20 }  
};
```

## Operações e Propriedades

Obtenção do comprimento do array:

`Length`

Copiando um array para outro:

`Clone()`

Pesquisa por um valor em um array

`Array.IndexOf(myArray, 3)`

Concatenação de dois arrays

`myArray1.Concat(myArray2).ToArray();`

Criar subconjuntos de um array usando o método

`Array.Slice()`

- Utilização de métodos de extensão LINQ, como: `Where`, `Select`, `Sum`, `Average`, etc

## Dictionary

O dicionário, ou dictionary, possui conceito similar a uma matriz de duas posições, pois é um tipo denominado de chave-valor, a qual é estruturada com dois tipos iguais ou diferentes para utilização, sendo o primeiro para determinar a chave e o segundo o valor, possuindo a estrutura Dictionary<tipo, tipo>.

```
Dictionary<string, string> gentilicos = new Dictionary<string, string>();
```

```
gentilicos.Add("SP", "Paulista");  
gentilicos.Add("PB", "Pernambucano");
```

## Estrutura de Repetição

São construções fundamentais em programação que permitem executar um bloco de código repetidamente com base em uma condição específica

## Estrutura de Repetição

```
while (condição) { // Código a ser repetido }
```

```
do { // Código a ser repetido } while (condição);
```

```
for (inicialização; condição; atualização) { // Código a ser repetido }
```

```
foreach (tipo elemento in coleção) { // Código a ser repetido para cada elemento na coleção }
```

Obrigado