C# - Exemplos práticos e explicações

Este documento contém exemplos simples e explicações curtas para cada tópico solicitado. Cada exemplo está formatado como código C# pronto para compilar em .NET 6+ (quando aplicável).

Hello World

Explicação: Programa mínimo que imprime "Hello, World!".

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello, World!");
    }
}
```

Variáveis e Tipos de Dados

Explicação: Declaração de variáveis primárias e atribuição.

```
using System;

class Program
{
    static void Main()
    {
        int idade = 30;
        double altura = 1.82;
        bool ativo = true;
        char inicial = 'C';
        string nome = "Caio";

        Console.WriteLine($"{nome}, {idade} anos, {altura}m, ativo: {ativo},
inicial: {inicial}");
    }
}
```

Operadores

Explicação: Aritméticos, relacionais e lógicos.

```
int a = 10, b = 3;
int soma = a + b;
int div = a / b; // inteiro
double divReal = (double)a / b;
bool igual = (a == b);
bool e = (a > 5) && (b < 5);

Console.WriteLine($"soma={soma}, div={div}, divReal={divReal}, igual={igual},
e={e}");
```

Estruturas de Controle (if, switch, loops)

Explicação: Condicionais e iterações.

```
// if / else
int x = 7;
if (x % 2 == 0)
    Console.WriteLine("par");
else
    Console.WriteLine("ímpar");

// switch
string dia = "terça";
switch (dia.ToLower())
{
    case "segunda": Console.WriteLine("Segunda"); break;
    case "terça": Console.WriteLine("Terça"); break;
    default: Console.WriteLine("Outro dia"); break;
}

// loops
for (int i = 0; i < 5; i++) Console.WriteLine(i);
int j = 0;
while (j < 3) { Console.WriteLine(j); j++; }

string[] nomes = { "Ana", "Bruno", "Carlos" };
foreach (var n in nomes) Console.WriteLine(n);
```

Métodos

Explicação: Definição e uso de métodos (funções).

```
using System;

class MathUtils
{
    public static int Somar(int x, int y) => x + y;
}

class Program
{
    static void Main()
    {
        int r = MathUtils.Somar(3, 4);
        Console.WriteLine(r); // 7
    }
}
```

Classes e Objetos

Explicação: Criação de classes, propriedades e instanciação.

```
public class Pessoa
{
    public string Nome { get; set; }
    public int Idade { get; set; }

    public void Apresentar() => Console.WriteLine($"Olá, sou {Nome}, {Idade}
anos.");
}

// Uso
var p = new Pessoa { Nome = "Maria", Idade = 28 };
p.Apresentar();
```

Herança

Explicação: Reuso de comportamento com `:` (herança simples).

```
public class Animal
{
    public string Nome { get; set; }
    public virtual void Falar() => Console.WriteLine("Som de animal");
}

public class Cachorro : Animal
```

```
{
    public override void Falar() => Console.WriteLine("Au au");
}

var c = new Cachorro { Nome = "Rex" };
c.Falar(); // Au au
```

Polimorfismo

Explicação: Usar uma referência de base para objetos derivados.

```
Animal a = new Cachorro();
a.Falar(); // chama override em Cachorro => Au au
```

Encapsulamento

Explicação: Controlar acesso a campos via propriedades.

```
public class Conta
{
    private decimal saldo;

    public decimal Saldo => saldo; // getter público, sem setter

    public void Depositar(decimal valor)
    {
        if (valor <= 0) throw new ArgumentException("Valor deve ser
positivo");
        saldo += valor;
    }
}

var conta = new Conta();
conta.Depositar(100);
Console.WriteLine(conta.Saldo);
```

Interfaces

Explicação: Contratos que classes implementam.

```
public interface IRepositorio<T>
{
```

```
    void Adicionar(T item);
    T? ObterPorId(int id);
}

public class RepositorioMemoria<T> : IRepositorio<T>
{
    private readonly List<T> _itens = new();
    public void Adicionar(T item) => _itens.Add(item);
    public T? ObterPorId(int id) => _itens.ElementAtOrDefault(id);
}
```

Exceções

Explicação: Tratamento com try/catch/finally e lançar exceções.

```
try
{
    int[] arr = {1,2};
    Console.WriteLine(arr[5]);
}
catch (IndexOutOfRangeException ex)
{
    Console.WriteLine("Índice inválido: " + ex.Message);
}
finally
{
    Console.WriteLine("Fim do bloco try");
}

// Lançar exceção
void Validar(int n)
{
    if (n < 0) throw new ArgumentOutOfRangeException(nameof(n));
}
```

Coleções (List, Dictionary)

Explicação: Uso de listas e dicionários.

```
var lista = new List<string> { "maçã", "banana" };
lista.Add("laranja");
Console.WriteLine(lista.Count);

var mapa = new Dictionary<string, int>();
mapa["um"] = 1;
```

```
mapa.Add("dois", 2);
Console.WriteLine(mapa["um"]);
```

Delegates e Events

Explicação: Delegates apontam para métodos; events são delegados protegidos.

```
public delegate void Notificador(string mensagem);

public class Publicador
{
    public event Notificador? AoNotificar;

    public void Disparar(string msg) => AoNotificar?.Invoke(msg);
}

var pub = new Publicador();
pub.AoNotificar += (m) => Console.WriteLine("Recebi: " + m);
pub.Disparar("Olá evento");
```

LINQ

Explicação: Consultas sobre coleções com `System.Linq`.

```
using System.Linq;

var numeros = new[] {1,2,3,4,5,6};
var pares = numeros.Where(n => n % 2 == 0).Select(n => n * 10);
foreach (var p in pares) Console.WriteLine(p); // 20,40,60

// Query syntax
var q = from n in numeros
        where n > 3
        select n;
```

Async/Await

Explicação: Programação assíncrona com Tasks.

```
using System;
using System.Threading.Tasks;
```

```
class Program
{
    static async Task<string> BaixarAsync()
    {
        await Task.Delay(500); // simula operação I/O
        return "conteúdo";
    }

    static async Task Main()
    {
        string texto = await BaixarAsync();
        Console.WriteLine(texto);
    }
}
```

Fim dos exemplos.