BATERRIA DE EXERCICIOS DAS AULA 28 29 COM GABARRITO

EXERCÍCIO 1: VARIÁVEIS E TIPOS DE DADOS

Descrição: Crie um programa que declare diferentes tipos de variáveis, faça algumas operações com elas e exiba os resultados.

CÓDIGO

```
using System;

class Program
{
    static void Main(string[] args)
    {
        // Declaração de variáveis
        int idade = 25;
        double altura = 1.75;
        bool estaEstudando = true;
        string nome = "João";

        // Operações com variáveis
        int anoNascimento = DateTime.Now.Year - idade;
        double alturaMetros = altura;
        string statusEstudo = estaEstudando ? "estudando" : "não estudando";

        // Exibição dos resultados
        Console.WriteLine($"Nome: {nome}");
        Console.WriteLine($"Idade: {idade}");
        Console.WriteLine($"Altura: {alturaMetros} metros");
        Console.WriteLine($"Ano de Nascimento: {anoNascimento}");
        Console.WriteLine($"Status de Estudo: {statusEstudo}");
    }
}
```

Explicação: O programa declara variáveis de tipos diferentes (int, double, bool, string), faz operações simples e exibe os resultados no console.

EXERCÍCIO 2: CLASSES E OBJETOS

Descrição: Crie uma classe Pessoa com propriedades para nome e idade. Instancie um objeto dessa classe e exiba suas propriedades.

CÓDIGO

```
using System;

class Pessoa
{
    public string Nome { get; set; }
    public int Idade { get; set; }
    public void ExibirInformacoes()
    {
        Console.WriteLine($"Nome: {Nome}");
        Console.WriteLine($"Idade: {Idade}");
    }
}


class Program
{
    static void Main(string[] args)
    {
        // Criação de um objeto Pessoa
        Pessoa pessoa = new Pessoa
        {
            Nome = "Maria",
            Idade = 30
        };
```

```
        // Exibe as informações da pessoa
        pessoa.ExibirInformacoes();
    }
}
```

Explicação: A classe Pessoa tem propriedades Nome e Idade, além de um método para exibir essas informações. O objeto é criado e suas propriedades são exibidas.

EXERCÍCIO 3: MÉTODOS E PARÂMETROS

Descrição: Crie um método que receba dois números inteiros como parâmetros e retorne a soma deles. Chame esse método e exiba o resultado.

CÓDIGO

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int numero1 = 10;
        int numero2 = 20;
        int resultado = Somar(numero1, numero2);
        Console.WriteLine($"A soma de {numero1} e {numero2} é {resultado}");
    }

    static int Somar(int a, int b)
    {
        return a + b;
    }
}
```

Explicação: O método Somar recebe dois parâmetros e retorna a soma deles. O resultado é exibido no console.

EXERCÍCIO 4: CLASSES COM CONSTRUTORES

Descrição: Crie uma classe Carro com um construtor que inicializa as propriedades Modelo e Ano. Instancie um objeto dessa classe e exiba suas propriedades.

CÓDIGO

```
using System;

class Carro
{
    public string Modelo { get; set; }
    public int Ano { get; set; }

    // Construtor
    public Carro(string modelo, int ano)
    {
        Modelo = modelo;
        Ano = ano;
    }
    public void ExibirInformacoes()
    {
        Console.WriteLine($"Modelo: {Modelo}");
        Console.WriteLine($"Ano: {Ano}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Criação de um objeto Carro
        Carro carro = new Carro("Fusca", 1975);
```

```
        // Exibe as informações do carro
        carro.ExibirInformacoes();
    }
}
```

Explicação: A classe Carro tem um construtor que inicializa Modelo e Ano. O objeto é criado usando o construtor e as propriedades são exibidas.

EXERCÍCIO 5: PROGRAMAÇÃO ASSÍNCRONA SIMPLES

Descrição: Crie um método assíncrono que simula um atraso de 2 segundos e retorna uma string. Chame esse método e exiba o resultado.

CÓDIGO

```
using System;
using System.Threading.Tasks;
class Program
{
    static async Task Main(string[] args)
    {
        string resultado = await SimularAtrasoAsync();
        Console.WriteLine(resultado);
    }
    static async Task<string> SimularAtrasoAsync()
    {
        await Task.Delay(2000); // Simula um atraso de 2 segundos
        return "Atraso concluído!";
    }
}
```

Explicação: O método SimularAtrasoAsync usa Task.Delay para simular um atraso e retorna uma string. O método Main aguarda a conclusão e exibe o resultado.

EXERCÍCIO 6: MANIPULAÇÃO ASSÍNCRONA DE ARQUIVOS

Descrição: Crie um método assíncrono que lê o conteúdo de um arquivo e o exibe. Use um arquivo de texto com algum conteúdo.

CÓDIGO

```
using System;
using System.IO;
using System.Threading.Tasks;
class Program
{
    static async Task Main(string[] args)
    {
        string caminhoArquivo = "exemplo.txt";
        string conteudo = await LerArquivoAsync(caminhoArquivo);
        Console.WriteLine($"Conteúdo do arquivo:\n{conteudo}");
    }

    static async Task<string> LerArquivoAsync(string caminhoArquivo)
    {
        return await File.ReadAllTextAsync(caminhoArquivo);
    }
}
```

Explicação: O método LerArquivoAsync lê o conteúdo de um arquivo de forma assíncrona e o exibe.

EXERCÍCIO 7: PROGRAMAÇÃO ASSÍNCRONA COM MÚLTIPLAS TAREFAS

Descrição: Crie um programa que executa duas tarefas assíncronas simultaneamente e aguarda a conclusão de ambas antes de exibir uma mensagem.

CÓDIGO

```
using System;
using System.Threading.Tasks;
class Program
{
    static async Task Main(string[] args)
    {
        Task tarefa1 = Atraso(1000, "Tarefa 1 concluída.");
        Task tarefa2 = Atraso(1500, "Tarefa 2 concluída.");

        await Task.WhenAll(tarefa1, tarefa2);
        Console.WriteLine("Ambas as tarefas foram concluídas.");
    }

    static async Task Atraso(int milissegundos, string mensagem)
    {
        await Task.Delay(milissegundos);
        Console.WriteLine(mensagem);
    }
}
```

Explicação: O método Atraso simula um atraso e exibe uma mensagem. O método Main executa duas tarefas simultaneamente e aguarda sua conclusão.

EXERCÍCIO 8: TRABALHANDO COM LISTAS ASSÍNCRONAS

Descrição: Crie um método assíncrono que recebe uma lista de números, simula um atraso para cada número e retorna a lista de números processados.

CÓDIGO

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

class Program
{
    static async Task Main(string[] args)
    {
        var numeros = new List<int> { 1, 2, 3, 4, 5 };
        var resultados = await ProcessarNumerosAsync(numeros);

        Console.WriteLine("Números processados:");
        foreach (var numero in resultados)
        {
            Console.WriteLine(numero);
        }
    }

    static async Task<List<int>> ProcessarNumerosAsync(List<int> numeros)
    {
        var tarefas = numeros.Select(async numero =>
        {
            await Task.Delay(500); // Simula um atraso de 500 ms
            return numero * 2; // Processa o número (exemplo: multiplica por 2)
        });

        return (await Task.WhenAll(tarefas)).ToList();
    }
}
```

Explicação: O método ProcessarNumerosAsync processa cada número da lista com um atraso e retorna a lista de números processados.