AQUI ESTÃO ALGUNS EXEMPLOS BÁSICOS DE COMO USAR ASYNC E AWAIT EM C#:

EXEMPLO 1: OPERAÇÃO ASSÍNCRONA SIMPLES

Vamos começar com um exemplo simples de um método assíncrono que simula uma operação que leva algum tempo para ser concluída.

```
using System;
using System.Threading.Tasks;
class Program
{
    static async Task Main(string[] args)
    {
        Console.WriteLine("Iniciando tarefa...");

        // Chama o método assíncrono
        await MetodoAssincrono();
        Console.WriteLine("Tarefa concluída.");
    }
    static async Task MetodoAssincrono()
    {
        // Simula uma operação de I/O, como uma chamada de API ou leitura de arquivo
        await Task.Delay(2000); // Aguarda por 2 segundos
        Console.WriteLine("Método assíncrono concluído.");
    }
}
```

EXPLICAÇÃO:

MetodoAssincrono é um método assíncrono que simula uma operação de 2 segundos.

await Task.Delay(2000) faz com que a execução do método aguarde por 2 segundos sem bloquear a thread principal.

O método Main também é assíncrono e usa await para aguardar a conclusão de MetodoAssincrono.

EXEMPLO 2: REQUISIÇÃO HTTP ASSÍNCRONA

Agora, vamos fazer uma requisição HTTP assíncrona usando HttpClient.

```
using System;
using System.Net.Http;
using System.Threading.Tasks;
class Program
{
    static async Task Main(string[] args)
    {
        Console.WriteLine("Iniciando requisição HTTP...");

        // Chama o método assíncrono para fazer a requisição
        string resultado = await
FazerRequisicaoHttpAsync("https://api.github.com/");
        Console.WriteLine("Resposta recebida.");
        Console.WriteLine(resultado);
    }
    static async Task<string> FazerRequisicaoHttpAsync(string url)
    {
        using HttpClient client = new HttpClient();

        // Configura o cabeçalho User-Agent para evitar erro 403
        client.DefaultRequestHeaders.Add("User-Agent", "C# App");

        // Faz a requisição HTTP GET de forma assíncrona
        HttpResponseMessage resposta = await client.GetAsync(url);
```

```
        // Lê o conteúdo da resposta como string
        string conteudo = await resposta.Content.ReadAsStringAsync();
        return conteudo;
    }
}
```

EXPLICAÇÃO:

O método FazerRequisicaoHttpAsync faz uma requisição HTTP assíncrona para um URL fornecido.

HttpClient.GetAsync(url) realiza a requisição HTTP GET.

await resposta.Content.ReadAsStringAsync() lê o conteúdo da resposta de forma assíncrona.

EXEMPLO 3: PROCESSAMENTO ASSÍNCRONO COM RESULTADOS

Vamos criar um exemplo que processa uma lista de números de forma assíncrona e calcula a soma.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
class Program
{
    static async Task Main(string[] args)
    {
        var numeros = Enumerable.Range(1, 10).ToList();
        Console.WriteLine("Iniciando processamento assíncrono...");

        // Chama o método assíncrono para processar a lista de números
        int soma = await ProcessarNumerosAssincronamente(numeros);
        Console.WriteLine($"Soma dos números: {soma}");
    }
```

```
    static async Task<int> ProcessarNumerosAssincronamente(List<int>
numeros)
    {
        // Simula o processamento assíncrono dos números
        var tarefas = numeros.Select(async numero =>
        {
            // Simula um trabalho com delay
            await Task.Delay(100);
            return numero;
        });

        // Aguarda a conclusão de todas as tarefas e calcula a soma
        var resultados = await Task.WhenAll(tarefas);
        return resultados.Sum();
    }
}
```

EXPLICAÇÃO:

O método ProcessarNumerosAssincronamente simula o processamento de uma lista de números com um pequeno atraso.

Task.WhenAll(tarefas) aguarda a conclusão de todas as tarefas geradas.

A soma dos resultados é calculada após a conclusão de todas as tarefas.