C# Avançado
Aula 07

- Revisão
- Exceções
- Resolver exercício
- Exercícios

C# Avançado - Arquivos

- try e Exceptions https://learn.microsoft.com/pt-br/dotnet/api/system.exception?view=net-8.0
- O bloco **try** deve ser seguido por um ou mais blocos **catch** e/ou um bloco **finally**, ou ambos. O bloco catch é executado quando ocorre um erro no bloco try. O bloco finally é executado após a saída da execução do bloco try (ou, se presente, do bloco catch) para realizar tarefas de limpeza, independentemente de ter ocorrido ou não uma exceção.

  Um bloco catch tem acesso a um objeto Exception que contém informações sobre o erro. Você utiliza um bloco catch para compensar o erro ou relançar a exceção. Você relança uma exceção se deseja apenas registrar o problema ou se deseja lançar um novo tipo de exceção de nível superior.

  Um bloco finally adiciona determinismo ao seu programa: o CLR se esforça para sempre executá-lo. É útil para tarefas de limpeza, como fechar conexões de rede.

  try { } // Onde executa o código

  catch (ExceptionA ex){ } //Onde trata erro do tipo específico que tentou no try

  catch (Exception ex){ } // Trata qualquer tipo de exceção

  catch (Exception ex){ throw ex} // Trata qualquer tipo de exceção e lança ela novamente para o código superior

  finally { }// Sempre executa mesmo com erro

C# Avançado - Arquivos

- Implementando exceções personalizadas

```
public class ExcecaoPersonalizadaException : Exception
{
    public ExcecaoPersonalizadaException (string mensagem) : base(mensagem)
    {
    }
}
```

C# Avançado - Exercícios

Neste exercício, você pode simular um programa que lida com a divisão de dois números, mas deve tratar exceções caso ocorra uma divisão por zero.

```
using System;
class Program{
    static void Main()    {
        Console.WriteLine("Bem-vindo ao programa de divisão!");
        try       {
            // Solicita ao usuário dois números
            Console.Write("Digite o numerador: ");
            int numerador = int.Parse(Console.ReadLine());
            Console.Write("Digite o denominador: ");
            int denominador = int.Parse(Console.ReadLine());
            // Realiza a divisão e exibe o resultado
            double resultado = RealizarDivisao(numerador, denominador);
            Console.WriteLine($"Resultado da divisão: {resultado}");
        }
        catch (FormatException)       {
            Console.WriteLine("Erro: Certifique-se de digitar números válidos.");
        }       catch (DivideByZeroException)       {
            Console.WriteLine("Erro: Divisão por zero não é permitida.");
        }       catch (Exception ex)       {
            Console.WriteLine($"Erro inesperado: {ex.Message}");
        }
        finally       {
            Console.WriteLine("O programa foi encerrado.");
        }
    }
    static double RealizarDivisao(int numerador, int denominador)
    {
        return (double)numerador / denominador;
    }
}
public class DivideByZeroException: Exception{
    public DivideByZeroException(string mensagem) : base(mensagem)
    {
    }
}
```

C# Avançado - Exercícios

Agora para praticar o uso de exceções, desta vez focando em exceções personalizadas e no uso de uma estrutura de controle de arquivos garantir a liberação de recursos.

```
using System;
class Program{
   static void Main()   {
      Console.WriteLine("Bem-vindo ao programa de manipulação de arquivos!");
      try      {
         // Solicita ao usuário o caminho do arquivo a ser lido
         Console.Write("Digite o caminho do arquivo: ");
         string caminhoArquivo = Console.ReadLine();
         // Tenta ler o conteúdo do arquivo
         string conteudo = LerArquivo(caminhoArquivo);
         // Exibe o conteúdo do arquivo
         Console.WriteLine($"Conteúdo do arquivo:\n{conteudo}");
      }
      catch (ArquivoNaoEncontradoException ex)      {
         Console.WriteLine($"Erro: {ex.Message}");
      }   catch (Exception ex)      {
         Console.WriteLine($"Erro inesperado: {ex.Message}");
      }
      finally      {
         Console.WriteLine("O programa foi encerrado.");
      }
   }
   static string LerArquivo(string caminho)
   {
    // Resolver
   }
}
public class ArquivoNaoEncontradoException : Exception
{
   public ArquivoNaoEncontradoException(string mensagem) : base(mensagem)
   {
   }
}
```