RESIDÊNCIA TIC 18

QUESTIONÁRIO DE NIVELAMENTO

Residente: Alessandro Conceição Santos

2. Sintaxe Básica de C#: Variáveis, Tipos de Dados e Operadores:

2.1. Explique a importância das variáveis em C# e forneça um exemplo de declaração.

Resposta: São fundamentais para armazenar e manipular dados durante a execução do programa.

Exemplo: int idade = 30;

2.2. Quais são os tipos de dados básicos em C# e como são utilizados?

Resposta: int, float, double, long, shor, char e bool. Podemos criar variáveis como no exemplo 2.1.

2.3. Dê exemplos de operadores aritméticos e lógicos em C#.

Resposta: Operadores aritméticos: +, - , /, *, %

Operadores lógicos: &&, ||,!

3. Estruturas de Controle de Fluxo: Condicionais e Loops em C#:

3.1. Como as estruturas condicionais são implementadas em C#? Dê um exemplo.

Resposta:

```
int idade = 20;

if (idade >= 18)

{

   Console.WriteLine("Você é maior de idade.");

}

else
```

```
{
    Console.WriteLine("Você é menor de idade.");
}
```

3.2. Explique o funcionamento dos loops em C# e forneça um exemplo de uso.

Resposta: Eles permitem executar repetidamente um trecho de código até que uma determinada condição seja verdadeira.

Exemplo:

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine("Número: " + i);
}
```

3.3. Qual é a diferença entre o "for" e o "while" em termos de controle de fluxo?

Reposta: For é utilizado quando se sabe quantas vezes o loop precisa ser executado.

While é utilizado quando não sabe quantas vezes o loop precisa ser executado.

4. Strings, Arrays e Listas, Datas:

4.1. Descreva operações comuns realizadas em strings em C#.

Resposta: Concatenação, pesquisa, substituição e manipulação de caracteres.

4.2. Compare e contraste Arrays e Listas em termos de funcionalidade e uso.

Resposta: Arrays tem tamanho fixo e so podem guardar um único tipo de dados.

Listas tem tamanho dinâmico e podem guardar diferentes tipos de dados.

4.3. Como as datas são representadas e manipuladas em C#? Dê exemplos.

Respostas: Elas são representadas pelo tipo **DATETIME**.

Exemplo:

```
DateTime dataAtual = DateTime.Now; // Obtém a data e hora atuais
Console.WriteLine("Data e hora atual: " + dataAtual);
```

```
DateTime dataFutura = dataAtual.AddDays(7); // Adiciona 7 dias à data atual

Console.WriteLine("Data daqui a uma semana: " + dataFutura);
```

5. Language Integrated-Query (LINQ):

5.1. O que é o LINQ e qual é sua finalidade em C#?

Resposta: Funciona como uma extensão de C# que permite consultar dados em tipos de fontes diferentes, como coleções ou banco de dados. Acredito que sua finalidade seja trazer facilidade, praticidade e eficiência ao programador na hora de buscar por dados.

5.2. Forneça um exemplo prático de utilização do LINQ em uma coleção de dados.

Resposta:

```
List<int> numeros = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var numerosPares = from num in numeros

                   where num % 2 == 0

                   select num;

foreach (var numero in numerosPares)

{

   Console.WriteLine(numero);

}
```

5.3. Quais são as vantagens de usar o LINQ em comparação com abordagens convencionais?

Resposta: Facilidade de uso e reutilização de código.

6. Debugging e Exceções:

6.1. Descreva o processo de debugging em C# e mencione ferramentas úteis.

Reposta: Envolve a identificação e correção de erros no código, podemos usar o Visual Studio Debugger.

6.2. Qual é o papel das exceções em C#? Como são tratadas no código?

Resposta: Elas ocorrem durante a execução do programa e elas podem ser tratadas dentro de bloco try-catch.

6.3. Dê um exemplo de como utilizar a declaração "try-catch" para lidar com exceções.

Resposta:

```
try{
   int divisor = 0;
   int resultado = 10 / divisor; // Tentativa de divisão por zero
   Console.WriteLine("Resultado: " + resultado); // Esta linha não será executada
}catch (DivideByZeroException ex){
   Console.WriteLine("Ocorreu uma exceção: " + ex.Message);
}
```

7. Conceitos de POO em C#:

7.1. Explique o que é Programação Orientada a Objetos (POO) e sua importância em C#.

Resposta: Conceito onde trazemos objetos do mundo real para a programação.

7.2. Quais são os pilares da POO? Descreva cada um brevemente.

Encapsulamento, Herança e Polimorfismo.

7.3. Dê um exemplo prático de como um objeto é criado em C#.

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


// Criando um objeto da classe Pessoa
Pessoa pessoa1 = new Pessoa();
pessoa1.Nome = "João";
```

pessoa1.Idade = 30;

8. Encapsulamento, Construtores e Destrutores:

8.1. Por que o encapsulamento é considerado importante na Programação Orientada a Objetos?

Ajuda a proteger os dados de uma classe, controlando o acesso a eles.

8.2. Explique a função dos construtores e destrutores em uma classe C#.

Usados para inicializar objetos de uma classe.

8.3. Como você implementaria um construtor em uma classe?

```
class Carro
{
    private string modelo;
    private int ano;
    public Carro(string modelo, int ano)
    {
        this.modelo = modelo;
        this.ano = ano;
    }
}
```

9. Herança e Polimorfismo:

9.1. O que é herança em C# e como ela é aplicada?

Mecanismo pelo qual uma classe pode herdar características (métodos e propriedades) de outra classe.

9.2. Explique o conceito de polimorfismo e forneça um exemplo prático.

Permite que objetos de classes diferentes sejam tratados de maneira uniforme.

9.3. Quais são as vantagens da herança e do polimorfismo na programação orientada a objetos?

Reutilização de código e facilidade de manutenção.

10. Classes Abstratas e Interfaces:

10.1. Qual é a diferença entre uma classe abstrata e uma interface em C#?

Classe Abstrata: É uma classe que não pode ser instanciada diretamente e pode conter métodos abstratos (métodos sem implementação) e métodos concretos (com implementação).

Interface: É um contrato que define um conjunto de membros (métodos, propriedades, eventos) que uma classe deve implementar.

10.2. Em que cenários você optaria por usar uma classe abstrata em vez de uma interface?

Quando fosse necessário uma implementação padrão para alguns métodos.

10.3. Dê um exemplo de implementação de uma interface em C#.

```
interface IImprimivel

{

   void Imprimir();

}

class Documento : IImprimivel

{

   public void Imprimir()

   {

      Console.WriteLine("Imprimindo o documento...");

   }

}

Documento doc = new Documento();

doc.Imprimir();
```