C# Fundamentals

Variáveis

Variáveis:

- As variáveis são utilizadas para armazenar informações na memória em tempo de execução da aplicação, isso significa que essas informações estarão disponíveis enquanto a aplicação estiver em execução e, mais precisamente, enquanto a classe ou método em que ela foi criada estiver em execução.

- No C# toda variável deve ter: modificador de acesso, tipo de dados e nome.

Variáveis:

- **Regras para nomes de variáveis:**
- **A regra para nomear uma variável é que o nome dela sempre comece por uma letra ou _. No meio do nome podem-se usar números, mas não se devem usar caracteres especiais e também não pode ser uma palavra reservada. Entende-se por palavras reservadas os comandos do C# e que são facilmente identificadas quando digitadas, por ficarem da cor azul. Exemplos de palavras que não podem ser utilizadas são: if, for, while, string e etc.**

Variáveis:

- **Modificadores:**

Modificador	Funcionamento
public	O acesso não é restrito.
protected	O acesso é limitado às classes ou tipos derivados daclasse que a variável está.
Internal	O acesso é limitado ao conjunto de módulos(assembly) corrente.
protected internal	O acesso é limitado ao conjunto corrente ou tiposderivados da classe recipiente.
private	O acesso é limitado à classe que a variável está.

Variáveis:

- **Tipos de dados:**
- **C# é uma linguagem de programação fortemente tipada, isso significa que todas as variáveis e objetos devem ter um tipo declarado. Os tipos de dados são divididos em value types e reference types. Os value types são derivados de System.ValueType e reference types são derivados de System.Object.**
- **O valor atribuído a uma variável deve estar de acordo com o seu tipo declarado. Por exemplo uma variável declarada com o tipo int não pode receber um valor numérico de ponto flutuante (de tipo double, float ou decimal).**

Variáveis:

- **Variáveis Value Type:**
- **As variáveis value type contém dentro delas um valor, enquanto as reference type contém uma referência. Isso significa que se copiar uma variável do tipo value type para dentro de outra o valor é copiado e, se o mesmo for feito com uma do tipo reference type será copiado apenas a referência do objeto.**
- **Dentro de Value Type existem duas categorias: struct e enum.**

Variáveis:

- **Struct:**
- **É dividida em tipos numéricos, bool e estruturas personalizadas pelo usuário.**

Tipo de dados	Intervalo
byte	0 ..255
sbyte	-128 ..127
short	-32,768 ..32,767
ushort	0 ..65,535
int	-2,147,483,648 ..2,147,483,647
uint	0 ..4,294,967,295
long	-9,223,372,036,854,775,808..9,223,372,036,854,775,807
ulong	0 ..18,446,744,073,709,551,615
float	-3.402823e38 ..3.402823e38
double	-1.79769313486232e308 ..1.79769313486232e308
decimal	-792281625142643375935439503 35..792281625142643375935439503 35
char	U+0000 .. U+ffff

Variáveis:

- **Enum:**
- **Permite criar um tipo que é formado por várias constantes. Normalmente é usada quando em algum momento existe a necessidade de um atributo que pode ter múltiplos valores, como por exemplo, em uma aplicação de venda que tem a tela de pedidos: cada pedido tem a sua situação que pode ser Aberto, Faturado e Cancelado.**

Variáveis:

- **Enum:**

```
public class Pedido
{
    public int numero;
    public DateTime dataHora;
    public ESituacao situacao;
}

public enum ESituacao
{
    Aberto = 1,
    Faturado = 2,
    Cancelado = 3
}
```

Variáveis:

- **Valores padrão:**
- **Todo tipo de dado tem um valor padrão que será atribuído a uma variável sem valor. Vejamos os principais casos de valor padrão:**
- **Os tipos inteiros (int) e de ponto flutuante (double, float, decimal) tem o valor padrão 0.**
- **Tipo bool representa um valor que pode ser verdadeiro ou falso, o seu valor padrão é false.**

Variáveis:

- **Operador new:**
- **Ao declarar uma nova variável podemos atribuir a instância de um tipo que preenche a variável com seu valor padrão.**
- **Podemos ver um exemplo do uso do operador new na Código 5.**
- **var x = new int(); // a variável x recebe o valor padrão 0**

Variáveis:

- **Variáveis Reference Types:**
- **As variáveis reference type armazenam apenas a referência do objeto. Os tipos de referência são: class, interface, delegate, object, string e Array.**
- **Tipo object: todos os tipos são derivados da classe Object, sendo assim é possível converter qualquer tipo para object.**
- **Tipo string: é utilizado para se armazenar caracteres e uma string deve estar entre aspas**
- **Exemplo: string nome = “Lucas";**

Variáveis:

- **Tipo string:**
- **Para concatenar (juntar) uma ou mais strings é usando o sinal de +**

```
string a = "C#";
string b = ".net";
string c = a + b;
```

Variáveis:

- **Tipo string:**
- **Outro recurso também é a possibilidade dese obter um determinado caractere de uma string**
- **A variável “d” vai ficar com o valor da letra “n” que é a letra que está no índice 3**

```
string a = "C#";
string b = ".net";
string c = a + b;
char d = c[3];
```

Variáveis:

- **Tipo string:**
- **Também é possível fazer a mesma coisa com a palavra**

```
char d = "C#.net"[3];
```

- **Para se obter o tamanho de uma variável do tipo string deve-se usar a propriedade Length. Com essa propriedade se obtém o número de caracteres de uma string.**

```
string palavra = “C#.net";
Console.WriteLine(palavra.Length); // 6
```

Variáveis:

- **Tipo array:**
- **Utilizado ao trabalhar com coleções de dados que possuem um tamanho previamente definido, armazenam múltiplas variáveis de um mesmo tipo.**
- **A grande vantagem do array está no fato de serem estruturas indexadas, ou seja, os itens do array podem ser acessados através de índices.**
- **A declaração padrão de um array obedece a seguinte estrutura:**
- **tipo[] nomeDoArray = {item, item, item};**

Variáveis:

- **Tipo array:**
- **Adicionado colchetes [] após o tipo da variável e o seu conteúdo fica entre chaves {}, separado por vírgula ,.**
- **Por exemplo, para um array do tipo string contendo formas geométricas, a declaração seria da seguinte forma:**
- **string[] array = {"Quadrado", "Círculo", "Triângulo", "Retângulo"};**

Variáveis:

- **Tipo array:**
- **Agora para obter algum dos itens armazenados no array, buscamos o seu valor pelo seu índice no array.**

```
string[] array = {"Quadrado", "Círculo", "Triângulo", "Retângulo"};
string forma =  array[2];
```

- **Lembrando que um índice começa do zero, no exemplo acima a variável forma irá buscar o valor do terceiro índice do array, retornando o valor "Triângulo".**

Variáveis:

- **Tipo array:**
- **Agora para obter algum dos itens armazenados no array, buscamos o seu valor pelo seu índice no array.**

```
string[] array = {"Quadrado", "Círculo", "Triângulo", "Retângulo"};
string forma =  array[2];
```

- **Lembrando que um índice começa do zero, no exemplo acima a variável forma irá buscar o valor do terceiro índice do array, retornando o valor "Triângulo".**

Variáveis:

- **Tipo array:**
- **Para se obter o tamanho de um array deve-se usar a propriedade Length. Com essa propriedade se obtém o número de índices em um array.**

```
string[] frutas = {“Laranja”, “Pera”, “Banana”};
Console.WriteLine(frutas.Length); //3
```


Dúvidas?