Apresentação

Curso:

Programação .NET I

Aula 05:

Objetivos:

Estruturas de tipo (*struct*);

Campos, propriedades e métodos;

Sobrecarga de operadores;

Exercício Prático (WindowsForm)

Exercício 1:
Name: pessoa_form
Solution name: Add to solution

Crie um formulário para o usuário entrar com dados de “nome” e “idade”, para instanciar um objeto do tipo Pessoa (como foi criado no Exercício 1) a partir do clique de um Botão.

O programa deve apresentar as propriedades do objeto no MessageBox padrão.

Tipo-estrutura comum (struct)

O struct nos permite criar uma estrutura de dados do tipo-valor. O tipo-valor é diferente do tipo-referência (classes). Quando se realiza uma cópia, copia-se o valor, enquanto que no caso de classes se copia a referência (tipo-referência).

Uma struct não permite heranças (é fixa com a System.ValueType).

Tipo-estrutura comum (struct)

O armazenamento dessas estruturas se dá pela pilha. Isso reduz a sobrecarga do gerenciamento da memória. (retira trabalho do GarbageCollector)

As estruturas possuem seus próprios campos, métodos e construtores, exatamente como se faz numa classe.

Tipo-estrutura comum (struct)

Quando utilizar uma struct?

- Quando as instâncias são pequenas e possuem um tempo de vida curto;
- Quando é necessário para se acoplar à um objeto (classe "*reference-type*");
- Quando NÃO representa um valor único (como int, double);
- Quando a premissa é não ser encapsulado frequentemente.

Tipo-estrutura comum (struct)

Estrutura de uma struct:

```
struct NomeDaStruct
    {
        tipo campo1, campo2;

         public tipo PropCampo1
        {
            get { return this.campo1; }
            set { this.campo1 = value; }
        }

        public tipo PropCampo2
        {
            get { return this.campo2; }
            set { this.campo2 = value; }
        }

         public tipo MetodoUm()
        {
              return "MetodoUm acionado!";
         }
    }
```

Tipo-estrutura comum (struct)

Instanciando um struct:

```
class Program
    {
        static void Main(string[] args)
        {

            NomeStruct nomeObjeto = new NomeStruct();
            nomeObjeto.PropCampo1 = XXXXX;

            Console.WriteLine(complexo.MetodoUm());
            Console.ReadKey(true);
        }
    }
```

Tipo-estrutura comum (struct)

Apontamentos de uma struct:

- O construtor padrão é sempre criado (diferente das Classes);
- Os campos precisam ser inicializados (diferente das Classes);
- Se você quiser criar um construtor NÃO padrão (que receba parâmetros), é necessário garantir que os campos serão inicializados primeiro (para isso, o Construtor Padrão deve ser chamado, pois é ele quem inicializa todos os campos. Para isso, utilize a palavra *this()* estendendo o construtor).
- Pode-se utilizar (e é recomendado pela documentação oficial) o "Inicializador de objetos" (*Object Initializer*), passando valores para as propriedades.
- A escrita das propriedades podem ser feitas após a declaração (instanciamento) e acessando pelo operador "."(ponto), atribuindo-lhe valor.

Tipo-estrutura comum (struct)

Utilizando construtor NÃO padrão:

```
struct NomeDaStruct
    {
        tipo campo1, campo2;

         public tipo PropCampo1
        {
            get { return this.campo1; }
            set { this.campo1 = value; }
        }

        public tipo PropCampo2
        {
            get { return this.campo2; }
            set { this.campo2 = value; }
        }

         public tipo NomeDaStruct (tipo variavel1, tipo variavel2):this();
        {
              this.PropCampo1 = variavel1;
              this.PropCampo2 = variavel2;
         }
    }
```

Tipo-estrutura comum (struct)

Instanciando um struct com construtor NÃO padrão:

```
class Program
    {
        static void Main(string[] args)
        {
            NomeStruct nomeObjeto = new NomeStruct(valor1, valor2);
        }
    }
```

Tipo-estrutura comum (struct)

Utilizando o inicializador de objetos:

```
struct NomeDaStruct
    {
        tipo campo1, campo2;

         public tipo PropCampo1
        {
            get { return this.campo1; }
            set { this.campo1 = value; }
        }

        public tipo PropCampo2
        {
            get { return this.campo2; }
            set { this.campo2 = value; }
        }




           ...
    }
```

Tipo-estrutura comum (struct)

Instanciando um struct com construtor NÃO padrão:

```
class Program
    {
        static void Main(string[] args)
        {
            NomeStruct nomeObjeto = new NomeStruct{PropCampo1 = valor1,
          PropCampo2 = valor2);
        }
    }
```

Tipo-estrutura comum (struct)

Utilizando o inicializador de objetos:

```
struct NomeDaStruct
    {
        tipo campo1, campo2;

         public tipo PropCampo1
        {
            get { return this.campo1; }
            set { this.campo1 = value; }
        }

        public tipo PropCampo2
        {
            get { return this.campo2; }
            set { this.campo2 = value; }
        }




           ...
    }
```

Tipo-estrutura comum (struct)

Instanciando um struct com construtor NÃO padrão:

```
class Program
    {
        static void Main(string[] args)
        {
            NomeStruct nomeObjeto = new NomeStruct();
            nomeObjeto.PropCampo1 = valor1;
            nomeObjeto.PropCampo2 = valor2;
        }
    }
```

Aplicação para struct

Imagine um problema em que você precise trabalhar com números complexos.

Uma utilização de struct pode ser uma solução bastante elegante.

Sabe-se que os números complexos possuem duas parcelas (real (a) e imaginária (b)) e a notação é dada por: “a + bi”

Tipo-estrutura comum (struct)

Exercício 2a:

Adicione à Solution uma nova Application para implementar uma struct de números complexos.

Dê o nome da struct de “NumeroComplexo”.

Template: C#; Windows; Windows Form

Name: “estruturas_comuns_struct”

Solution: “Add to solution”

Resposta de exercício

```
struct NumeroComplexo
    {
        private double pReal;
        private double pImag;


        public double PImag
        {
            get { return this.pImag; }
            set { this.pImag = value; }
        }

        public double PReal
        {
            get { return this.pReal; }
            set { this.pReal = value; }
        }
    }
```

Tipo-estrutura comum (struct)

Exercício 2b:

Na struct criada, sobrecarregue o método ToString() para que o número possa ser impresso obedecendo a notação: “a + bi”

a = Parte Real
b = Parte Imaginária

Resposta de exercício

```
struct NumeroComplexo
    {
        private double pReal;
        private double pImag;


        public double PImag
        {
            get { return this.pImag; }
            set { this.pImag = value; }
        }

        public double PReal
        {
            get { return this.pReal; }
            set { this.pReal = value; }
        }

        public override string ToString()
        {
            return this.PReal + " + " + this.PImag + "i";
        }

    }
```

Tipo-estrutura comum (struct)

Exercício 3c:

Na mesma Application, utilize dois TextBox (um para a parte real e outro para a parte imaginária) e um Button para instanciar um objeto do tipo NumeroComplexo e apresenta-lo num Label1.

Resumo de aula

- Mandem os exercícios de aula para a tarefa no moodle.