INTRODUÇÃO AO FRAMEWORK .NET

Cairu

Fundação Visconde de Cairu - Desde 1905

AULA 3 – STRING E ARRAY

- Principais métodos
- Array e Listas
- Foreach
- Generics

APRESENTAÇÃO DIA 1 – 11/07 – PROF. ARISTÓTELES

Início	Fim	Atividade
18:30	19:20	Apresentação
19:20	20:00	Introdução ao Framework .Net e Apresentação do Visual Studio 2013 (github)
20:00	20:10	Intervalo
20:10	20:50	Introdução ao C#: tipos de dados, if, else, case, for, while, e/s console
20:50	21:30	Prática 1 – Console Application

STRING - LENGTH

- Retorna a quantidade de caractere da string

```
string str = "João Carlos";
int tam = str.Length();
```

STRING – TOUPPER E TOLOWER

- Retorna uma cópia da string com letras maiúsculas e minúsculas

```
string str = "João Carlos";
String strUpper = str.ToUpper();
string strLower = str.ToLower();
```

STRING – TRIM

- Retorna uma cópia da string sem o espaço em branco do início e do final

```
String str = "  João Carlos   ";
String strTrim = str.Trim();
```

STRING – SPLIT

- Divide uma string em várias de acordo com um delimitador e devolve um array com as strings obtidas

```
string str = "João, Maria, Vitor ";
string[] strSplit = str.Split(',');
```

STRING – REPLACE

- Cria uma cópia de uma strnig substituindo um parte por outro conteúdo

```
string str = "João, Maria, Vitor ";
string strReplace = str.Replace("Maria", "Carlos");
```

ARRAYS

Arrays de 1 Dimensão

```
decimal[] valoresDiarios = new decimal[366];
string[] Meses = new string[12];
```

Obs.: Primeiro elemento é o 0.

Arrays de 2 Dimensões

```
float[,] retangulo = new float[5, 10];
```

ARRAYS – OUTRAS FORMAS

```
 string[] semana = {"Segunda", "Terca",
"Quarta","Quinta","Sexta", "Sabado", "Domingo" };
```

```
System.DateTime[] Datas = new System.DateTime[4];
Datas[0] = Convert.ToDateTime("12/05/2002");
Datas[1] = Convert.ToDateTime("25/08/1965");
Datas[2] = Convert.ToDateTime("30/03/1978");
```

ARRAYS – OUTRAS FORMAS

```
int[] Salas1 = new int[16];
int[] Salas2 = new int[13];
int[] Salas3 = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

ARRAYS – PERCORRENDO FOR

```
int[] array = new int[3];
array[0] = 100;
array[1] = 10;
array[2] = 1;
for (int index = 0; index <= 3; index++)
{

     Console.WriteLine(array[index]);
}
```

ARRAYS – PERCORRENDO FOR EACH

foreach (variavel in conjunto){

instruções

}

```
int[] array = new int[3];
array[0] = 100;
array(1) = 10;
Array[2] = 1;
foreach (int element in array)
{

    Console.WriteLine(element);
}
```

ARRAYS – ORDENANDO

```
int[] numeros = { 3, 2, 1, 0 };
System.Array.Sort(numeros);
foreach (int element in numeros)
{
    Console.WriteLine(element);
}
```

COLLECTIONS

- Estruturas de Dados para manipular grandes quantidades de dados
- As estrutura de dados mais básica é array porém possui diversas limitações
  - Tamanho Fixo
  - Dificuldade de redimensionar
  - Deslocar conteúdo para adicionar e remover
- Necessidade de Estrutura de Dados mais sofisticadas (Listas)

LISTA - INSERT

- Adicionar elementos na lista em uma determinada posição

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
```

LISTA

- Interface Ilist define métodos que uma lista deve implementar
- A principal implementação é ArrayList

```
ArrayList lista = new ArrayList();
IList lista2 = new ArrayList();
```

LISTA - COUNT

- Informa a quantidade de elementos de uma lista

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
int qtd = lista.Count();
```

LISTA - CLEAR

- Remove todos os elementos da lista

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
lista.Clear();
```

LISTA - REMOVE

- Remove a primeira ocorrência do elemento passado como parâmetro

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
lista.Remove(123);
```

LISTA - REMOVEAT

- Remove o elemento do índice passado como parâmetro

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
lista.RemoveAt(1);
```

LISTA - ITEM

- Recupera o elemento do índice passado como parâmetro

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
string str = lista.Item(1);
```

LISTA - INDEXOF

- Descobre o índice do elemento do valor passado como parâmetro

```
IList lista = new ArrayList();
lista.Insert(0, "Maria");
lista.Insert(1, 123);
lista.Insert(1, 12.75);
int indice = lista.IndexOf("Maria");
```

GENERICS

- As listas armazenam referência de qualquer tipo, então para ter acesso aos métodos específicos dos objetos armazenados tem que se fazer o casting
- Existe o erro de haver casting errado causando erro em tempo de execução
- Com o recurso Generics podemos determinar o tipo de objeto que será armazenado na lista
- As classes List e LinkedList implementam o recurso Generics

```
List<int> lista = new List<int>();
LinkedList<int> lista2 = new LinkedList<int>();
```

PRÁTICA 3