ArrayList

Unidade I: Introdução às Collections

PUC Minas Virtual

Collections em C#

- Estruturas de dados pré-empacotadas na plataforma .NET
- Utilizando tais classes, o programador não preocupa com a maneira que as mesmas foram implementadas (reutilização de código)
- É necessário usar o *namespace System.Collections*
- Alguns exemplos: *ArrayList*, *List*, *LinkedList*, *Queue*, *Stack* e *Hashtable* e *Dictionary*

Classe *ArrayList*

- Similar ao *array*, ela também consegue redimensionar dinamicamente seu tamanho se necessário. O tamanho inicial do *array* é imutável
- Um *ArrayList* pode conter uma quantidade de elementos menor ou igual à sua capacidade (*Capacity*)
- Armazena referências para objetos de qualquer tipo

Propriedade *Capacity*

- Permite manipularmos a capacidade de um *ArrayList*
- Por padrão, seu valor inicial é zero e, após a inserção do primeiro elemento, torna-se quatro
- Por padrão, tem seu valor duplicado automaticamente quando o *ArrayList* precisa de crescer

Propriedade *Count*

- Quantidade de elementos no *ArrayList*

Criação de Objetos do Tipo *ArrayList*

```
//Criação sem informar a capacidade
ArrayList al1 = new ArrayList();

//Criação informando a capacidade
ArrayList al2 = new ArrayList(10);
```

Exercício Resolvido (1)

- Faça um programa contendo um objeto *ArrayList* e que imprime sua capacidade (*Capacity*) e quantidade de elementos (*Count*) na tela. Em seguida, insira um elemento usando o método *Add* e repita a impressão

Exercício Resolvido (1)

- Faça um programa contendo um objeto *ArrayList* e que imprime sua capacidade (*Capacity*) e quantidade de elementos (*Count*) na tela. Em seguida, insira um elemento usando o método *Add* e repita a impressão

Pause!

Exercício Resolvido (1)

```
using System;
using System.Collections;

class MainClass {
        public static void Main (string[] args) {
                ArrayList al = new ArrayList();
                Console.Write("AL.Capacity({0}) / AL.Count({1})\n", al.Capacity, al.Count);
                al.Add(1);
                Console.Write("AL.Capacity({0}) / AL.Count({1})\n", al.Capacity, al.Count);
        }
}
```

Comandos *foreach*

- Efetua a repetição como nos comandos *for* / *while* / *do*
- Exemplo:

```
ArrayList al = new ArrayList();
                …
for(int i = 0; i < al.Count(); i++)
        Console.WriteLine(al[i]);

foreach(object o in al)
        Console.WriteLine(o);

foreach(int num in al)
        soma = soma + num;
```

Exercício Resolvido (2)

- Faça um programa que leia 5 números inteiros, os adicione em um *ArrayList* e calcule a média deles. Em seguida, mostre na tela aqueles que forem maiores que a média. Seu programa terá duas estruturas de repetição sendo a primeira, um ***for***; e a segunda, um ***foreach***

Exercício Resolvido (2)

- Faça um programa que leia 5 números inteiros, os adicione em um *ArrayList* e calcule a média deles. Em seguida, mostre na tela aqueles que forem maiores que a média. Seu programa terá duas estruturas de repetição sendo a primeira, um ***for***; e a segunda, um ***foreach***

Pause!

Exercício Resolvido (2)

```
ArrayList al = new ArrayList();
double media = 0;
for (int i = 0; i < 5; i++){
        int valor = int.Parse(Console.ReadLine());
        al.Add(valor);
        media += valor;
}
media /= 5;
foreach (object o in al){
        if((int)o > media){
                Console.WriteLine(o);
        }
}
```

adicione em um
tre na tela aqueles
duas estruturas
, um ***foreach***

Alguns Métodos da Classe *ArrayList*

- **Add**: insere um objeto e retorna sua posição
- **Insert**: Insere um objeto na posição especificada. Ocorre uma exceção se a posição não existir
- **Remove**: remove a primeira ocorrência do objeto especificado
- **RemoveAt**: remove um objeto no índice especificado
- **RemoveRange**: remove uma determinada quantidade de elementos à partir do índice especificado

Alguns Exemplos da Aplicação de Métodos da Classe *ArrayList*

```
ArrayList al = new ArrayList();
                …
al.Add(15);
al.Add(3.14159);
al.Add("AEDs");
al.Insert(2, 125); // Adiciona 125 na posição 2
al.Remove(3.14159); // Não ocorre exceção se elemento inexistente
al.RemoveAt(1); // Ocorre exceção de posição inexistente
al.RemoveRange(0, 2); // Remove 2 elementos à partir da posição 0. Ocorre exceção
                      // se não existir a quantidade de elementos desejada
```

Alguns Métodos da Classe *ArrayList*

- **Clear**: remove todos os elementos sem alterar a capacidade
- **Contains**: retorna se a estrutura contém um objeto
- **IndexOf**: retorna o índice da primeira ocorrência de um dado objeto
- **LastIndexOf**: retorna o índice da última ocorrência de um dado objeto

Alguns Exemplos da Aplicação de Métodos da Classe *ArrayList*

```
ArrayList al = new ArrayList();
                …
al.Clear();
                …
if (al.Contains(15) == true){
        Console.Write("Elemento encontrado");
}
int quantidade = al.Count;
int primeiraPosicaoDo15 = al.IndexOf(15);
int ultimaPosicaoDo15 = al.LastIndexOf(15);
```

Alguns Métodos da Classe *ArrayList*

- **Reverse**: inverte a ordem dos elementos
- **Sort**: Ordena os objetos
- **ToArray**: Copia os objetos para um *array*
- **TrimToSize**: Altera a *Capacity* para a quantidade de elementos
- **BinarySearch**: Realiza uma busca binária e retorna a posição do elemento ou um número negativo se não for encontrado

Alguns Exemplos da Aplicação de Métodos da Classe *ArrayList*

```
ArrayList al = new ArrayList();
            …
al.Reverse(); // Inverte os elementos de todo o ArrayList
al.Reverse(3,5); // Inverte os 5 elementos à partir da posição 3
al.Sort();
Object[] vetor = al.ToArray();
al.TrimToSize();
int posicao = al.BinarySearch("AEDs");
```

Exercício Resolvido (3)

- Faça um programa que use todos os métodos aprendidos para *ArrayList*