Fundamentos de Desenvolvimento com C#

Aula 09: Coleções

Professor: Rinaldo Ferreira Junior

E-mail: rinaldo.fjunior@prof.infnet.edu.br

- **Professor:** Rinaldo Ferreira Junior
- **Graduação:** Pós-graduado em Arquitetura de Softwares
- **Atuação:** .Net | C# | SQL | NoSQL | Engenheiro de Software
- **E-mail:** rinaldo.fjunior@prof.infnet.edu.br
- **Linkedin:** https://www.linkedin.com/in/rinaldo-ferreira-junior-787326a

Aula 09: Roteiro

- Coleções

Aula 09: Coleções

Instituto Infnet

- Há duas formas de se manipular grupos de elementos em .Net, arrays e collections.
  - Em linhas gerais, arrays são usados para armazenar um conjunto fixo de dados, enquanto collections são usadas para conjuntos dinâmicos.
- Embora hajam diferentes tipos de coleção, elas podem compartilhar algumas funcionalidades.
  - Localizar elementos
  - Adicionar ou remover elementos dinamicamente
  - Listar elementos
- Os tipos de coleção mais comuns, se encontram nos namespaces System.Collections e System.Collections.Generic.

Aula 09: List<T>

Instituto Infnet

- Representa uma lista genérica, de algum tipo de dado.
- Seus membros são indexados a partir de 0.
- Podem conter elementos repetidos.
- Possui métodos para busca, ordenação e manipulação de seus membros.
- A capacidade da lista muda dinamicamente, a medida que novos membros são adicionados.
  - A capacidade inicial de uma lista é 0
  - Ao adicionar um elemento, a capacidade é dinamicamente aumentada para 4
  - Podem ser inicializadas já com uma capacidade inicial definida

Aula 09: Adicionando em Listas

- O método Add insere um único elemento à lista.
- O método AddRange insere múltiplos objetos à lista.

```
string? selecao = null;
string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
List<string> estados = new();
estados.AddRange(nomeEstados);
estados.Add("Rio Grande do Sul");
```

Aula 09: Removendo em Listas

- O método Remove remove um único elemento da lista.
- O método RemoveAt remove um único elemento da lista com base em seu índice.
- O método RemoveRange remove múltiplos elementos a partir de um índice.

```
estados.Remove("Minas Gerais");
estados.RemoveAt(1);
estados.RemoveRange(2, 3);
```

Aula 09: Busca em Listas

Instituto Infnet

- O método Find busca e retorna o primeiro elemento que corresponde ao predicado, ou o valor default do tipo.

```
static void Main(string[] args)
{
    string? selecao = null;
    string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
    List<string> estados = new();
    estados.AddRange(nomeEstados);
    estados.Add("Rio Grande do Sul");

    do
    {
        Console.Write("Informe o nome do Estado: ");
        selecao = Console.ReadLine();

    } while (string.IsNullOrWhiteSpace(selecao));

    Address endereco = new();

    endereco.Logradouro = "Rua Blah";
    endereco.Complemento = "Bloco 1";
    endereco.Bairro = "Centro";
    endereco.Estado = estados.Find(n => n == (!string.IsNullOrWhiteSpace(selecao) ? selecao : ""))!;

    Person pessoa = new Person();
    pessoa.Nome = "Roberval Firmino";
    pessoa.Endereco = endereco;
    pessoa.Nascimento = new DateTime(1965, 04, 04);

    Console.WriteLine($"Logradouro: {endereco.Logradouro} | Complemento: {endereco.Complemento}");
    Console.WriteLine($"Bairro: {endereco.Bairro} | Possui número? {(endereco.SemNumero ? "Não" : endereco.Numero)}");
    Console.WriteLine($"Estado: {endereco.Estado}");
    Console.WriteLine($"País: {endereco.Pais}");

    Console.Read();
}
```

Aula 09: Busca em Listas

Instituto Infnet

- O método First busca e retorna o primeiro elemento da lista que satisfaça aos critérios de busca.

```
static void Main(string[] args)
{
    string? selecao = null;
    string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
    List<string> estados = new();
    estados.AddRange(nomeEstados);
    estados.Add("Rio Grande do Sul");

    do
    {
        Console.Write("Informe o nome do Estado: ");
        selecao = Console.ReadLine();

    } while (string.IsNullOrWhiteSpace(selecao));

    Address endereco = new();

    endereco.Logradouro = "Rua Blah";
    endereco.Complemento = "Bloco 1";
    endereco.Bairro = "Centro";
    endereco.Estado = estados.FirstOrDefault(e => e.Contains(selecao, StringComparison.OrdinalIgnoreCase))!;

    Person pessoa = new Person();
    pessoa.Nome = "Roberval Firmino";
    pessoa.Endereco = endereco;
    pessoa.Nascimento = new DateTime(1965, 04, 04);

    Console.WriteLine($"Logradouro: {endereco.Logradouro} | Complemento: {endereco.Complemento}");
    Console.WriteLine($"Bairro: {endereco.Bairro} | Possui número? {(endereco.SemNumero ? "Não" : endereco.Numero)}");
    Console.WriteLine($"Estado: {endereco.Estado}");
    Console.WriteLine($"País: {endereco.Pais}");

    Console.Read();
}
```

Aula 09: Busca em Listas

- O método ElementAt busca e retorna o elemento na posição pedida.

```
static void Main(string[] args)
{
    int? selecao = null;
    string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
    List<string> estados = new();
    estados.AddRange(nomeEstados);
    estados.Add("Rio Grande do Sul");

    do
    {
        Console.Write("Informe a posição do Estado: ");
        selecao = Convert.ToInt32(Console.ReadLine());

    } while (selecao is null);

    Address endereco = new();

    endereco.Logradouro = "Rua Blah";
    endereco.Complemento = "Bloco 1";
    endereco.Bairro = "Centro";
    endereco.Estado = estados.ElementAt(selecao.Value);

    Person pessoa = new Person();
    pessoa.Nome = "Roberval Firmino";
    pessoa.Endereco = endereco;
    pessoa.Nascimento = new DateTime(1965, 04, 04);

    Console.WriteLine($"Logradouro: {endereco.Logradouro} | Complemento: {endereco.Complemento}");
    Console.WriteLine($"Bairro: {endereco.Bairro} | Possui número? {(endereco.SemNumero ? "Não" : endereco.Numero)}");
    Console.WriteLine($"Estado: {endereco.Estado}");
    Console.WriteLine($"País: {endereco.Pais}");

    Console.Read();
}
```

Aula 09: Busca em Listas

Instituto Infnet

- Buscas pelo indexador, agem como o método ElementAt.

```
static void Main(string[] args)
{
    int? selecao = null;
    string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
    List<string> estados = new();
    estados.AddRange(nomeEstados);
    estados.Add("Rio Grande do Sul");

    do
    {
        Console.Write("Informe a posição do Estado: ");
        selecao = Convert.ToInt32(Console.ReadLine());

    } while (selecao is null);

    Address endereco = new();

    endereco.Logradouro = "Rua Blah";
    endereco.Complemento = "Bloco 1";
    endereco.Bairro = "Centro";
    endereco.Estado = estados[selecao.Value];

    Person pessoa = new Person();
    pessoa.Nome = "Roberval Firmino";
    pessoa.Endereco = endereco;
    pessoa.Nascimento = new DateTime(1965, 04, 04);

    Console.WriteLine($"Logradouro: {endereco.Logradouro} | Complemento: {endereco.Complemento}");
    Console.WriteLine($"Bairro: {endereco.Bairro} | Possui número? {(endereco.SemNumero ? "Não" : endereco.Numero)}");
    Console.WriteLine($"Estado: {endereco.Estado}");
    Console.WriteLine($"Pais: {endereco.Pais}");

    Console.Read();
}
```

Aula 09: Iteração em Listas

Instituto Infnet

- Para coleções, a instrução foreach é mais indicada, pois manipula diretamente a instância de cada elemento da lista.

```
string? selecao = null;
string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
List<string> estados = new();
estados.AddRange(nomeEstados);
estados.Add("Rio Grande do Sul");

Console.WriteLine("O banco está nos seguintes estados:");
foreach (string item in estados)
{
    Console.WriteLine($"{estados.IndexOf(item)} - {item}");
}

Console.WriteLine();
do
{
    Console.Write("Informe o nome do Estado: ");
    selecao = Console.ReadLine();

} while (string.IsNullOrWhiteSpace(selecao));
```

Aula 09: Capacidade

Instituto Infnet

- A propriedade Capacity pré aloca espaço para os elementos que serão adicionados à uma lista.
- Quando Count for igual a Capacity, a lista é redimensionada automaticamente.
- O redimensionamento dobra a capacidade da lista

```
string[] nomeEstados = { "Rio de Janeiro", "Minas Gerais", "Santa Catarina", "São Paulo", "Paraná", "Bahia" };
List<string> estados = new(10);
estados.AddRange(nomeEstados);
estados.Add("Rio Grande do Sul");

Console.WriteLine($"Capacidade inicial: {estados.Capacity}");
Console.WriteLine($"Carga inicial: {estados.Count}");
estados.AddRange(new string[] { "Amazonas", "Pará", "Acre", "Roraima" });
Console.WriteLine($"Capacidade final: {estados.Capacity}");
Console.WriteLine($"Carga final: {estados.Count}");
Console.WriteLine();
```

```
Capacidade inicial: 10
Carga inicial: 7
Capacidade final: 20
Carga final: 11
```