Linguagem de Programação 3

Aula 9
Classes de Coleções

l.bertholdo@ifsp.edu.br

Conteúdo

- Classes de Coleção Genéricas
  - List <T>
  - Queue <T>
  - Stack <T>

l.bertholdo@ifsp.edu.br

Classes de Coleção Genéricas

- Arrays são muito úteis, porém têm algumas limitações:
  - Não é possível aumentar ou diminuir seu tamanho.
  - A ordenação e o rearranjo dos elementos de um array não é simples.
  - Seus índices precisam ser números inteiros para que os dados possam ser acessados.
- O .NET Framework possui classes específicas para implementar e manipular coleções de elementos.
- São as chamadas **classes de coleção genéricas**, disponibilizadas no namespace **System.Collections.Generic**.

Classes de Coleção Genéricas

- As coleções genéricas foram adicionadas à versão do C# desenvolvida para o .NET Framework 2.0.
- Essas coleções são consideradas genéricas pois não têm vínculo com nenhum tipo de dado. Ao declará-las, é preciso informar o tipo de dado que armazenarão. Exemplo: List<int>.
- Ao criar uma coleção genérica, seu tamanho não precisa ser especificado, já que ele aumenta ou diminui conforme os elementos são inseridos ou removidos da coleção.
- Cada classe de coleção é otimizada para uma forma específica de armazenamento e acesso aos dados. Por isso, cada classe possui seus próprios métodos e propriedades para realizar estas funções.

List<T>

- List<T> é a mais simples das classes de coleção, pois funciona de forma similiar a um array comum, com a diferença que a alocação dos elementos na memória é feita dinamicamente.
- Os elementos de uma coleção List<T> são armazenados de forma contígua na memória, igual a um array comum.
- Por isso, diferentemente de uma lista encadeada, a informação de quais são os elementos vizinhos é desnecessária.

l.bertholdo@ifsp.edu.br

List<T>

- Principais métodos:
  - **Add:** Permite adicionar um elemento ao final da lista, aumentando automaticamente seu tamanho.

    Sintaxe: <nome_lista>.Add(<valor>)

  - **Insert:** Permite adicionar um elemento em qualquer posição da lista, aumentando automaticamente seu tamanho.

    Sintaxe: <nome_lista>.Insert(<posição>, <valor>)

  - **Sort:** Permite ordenar os elementos da lista.

    Sintaxe: <nome_lista>.Sort()

* A classe List<T> pode ser inicializada com valores ao ser declarada. Exemplo:

```
List<string> nomes = new List<string>() {"João","Pedro","Paulo","Maria","Carlos"};
```

l.bertholdo@ifsp.edu.br

List<T>

- Principais métodos:
  - **Remove:** Permite remover qualquer elemento da lista, fechando a lacuna onde o elemento estava e mantendo a ordem dos elementos após a remoção.

    Sintaxe: <nome_lista>.Remove(<valor>)

  - **RemoveAt:** Semelhante ao método **Remove**, porém deve-se informar a posição do elemento a ser removido e não seu valor.

    Sintaxe: <nome_lista>.RemoveAt(<posição>)

l.bertholdo@ifsp.edu.br

List<T>

```
João, Pedro, Paulo, Maria, Carlos,
João, Pedro, Paulo, Maria, Ivo, Carlos,
João, Paulo, Maria, Ivo, Carlos,
João, Paulo, Ivo, Carlos,
Carlos, Ivo, João, Paulo,
```

```
class Lista{
    static void Main(string[] args){
        List<string> nomes = new List<string>();

        // Adiciona 5 elementos à lista "nomes".
        foreach (string n in new string[5] {"João","Pedro","Paulo","Maria","Carlos"})
            nomes.Add(n);
        foreach (string n in nomes) // Imprime a lista
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Insere o nome "Ivo" na penúltima posição da lista (nomes.Count - 1).
        nomes.Insert(nomes.Count - 1, "Ivo"); // Count retorna o número de elementos da lista.
        foreach (string n in nomes) // Imprime a lista
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Remove o 1º elemento cujo valor é "Pedro".
        nomes.Remove("Pedro");
        foreach (string n in nomes) // Imprime a lista
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Remove o elemento cuja posição é 2.
        nomes.RemoveAt(2);
        foreach (string n in nomes) // Imprime a lista
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Ordena a lista.
        nomes.Sort();
        foreach (string n in nomes) // Imprime a lista
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();
    }
}
```

ldo@ifsp.edu.br

Queue<T>

- A classe Queue<T> implementa o mecanismo de fila, onde o 1º elemento a entrar é o 1º a sair (*FIFO – first-in, first-out*).
- Principais métodos:
  - **Enqueue:** Insere o elemento no final da fila.
    Sintaxe: <nome_fila>.Enqueue(<valor>)
  - **Dequeue:** Remove o 1º elemento da fila.
    Sintaxe: <nome_fila>.Dequeue()

Queue<T>

```
João, Pedro, Paulo, Maria, Carlos,
João, Pedro, Paulo, Maria, Carlos, Ana,
Pedro, Paulo, Maria, Carlos, Ana,
```

```
class Fila{
    static void Main(string[] args){
        Queue<string> nomes = new Queue<string>();

        // Adiciona 5 elementos à fila "nomes".
        foreach (string n in new string[5] {"João","Pedro","Paulo","Maria","Carlos"})
            nomes.Enqueue(n);
        foreach (string n in nomes) // Imprime a fila
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Adiciona 1 elemento no final da fila "nomes".
        nomes.Enqueue("Ana");
        foreach (string n in nomes) // Imprime a fila
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Remove 1 elemento do início da fila "nomes".
        nomes.Dequeue();
        foreach (string n in nomes) // Imprime a fila
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Remove todos os elemento da fila "nomes".
        while (nomes.Count > 0) // Count retorna o número de elementos da fila.
            nomes.Dequeue();
        foreach (string n in nomes) // Imprime a fila
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();
    }
}
```

Stack<T>

- A classe Stack<T> implementa o mecanismo de pilha, onde o último elemento a entrar é o 1º a sair (*LIFO – last-in, first-out*).
- Principais métodos:
  - **Push:** Insere o elemento no topo da pilha.
    Sintaxe: <nome_pilha>.Push(<valor>)
  - **Pop:** Remove o elemento que está no topo da pilha.
    Sintaxe: <nome_pilha>.Pop()

l.bertholdo@ifsp.edu.br

Stack<T>

```
Carlos, Maria, Paulo, Pedro, João,
Maria, Paulo, Pedro, João,
Ana, Maria, Paulo, Pedro, João,
```

```
class Pilha{
    static void Main(string[] args){
        Stack<string> nomes = new Stack<string>();

        // Adiciona 5 elementos à pilha "nomes".
        foreach (string n in new string[5] {"João","Pedro","Paulo","Maria","Carlos"})
            nomes.Push(n);
        foreach (string n in nomes) // Imprime a pilha
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Remove o elemento que está no topo da pilha "nomes".
        nomes.Pop();
        foreach (string n in nomes) // Imprime a pilha
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Insere o elemento no topo da pilha "nomes".
        nomes.Push("Ana");
        foreach (string n in nomes) // Imprime a pilha
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();

        // Remove todos os elementos da pilha "nomes" a partir do topo.
        while (nomes.Count > 0) // Count retorna o número de elementos da pilha.
            nomes.Pop();
        foreach (string n in nomes) // Imprime a pilha
            Console.Write(n + ", "); Console.ReadKey(); Console.WriteLine();
    }
}
```

"Carlos" está no topo da pilha, por isso ele é o primeiro a ser impresso.

Agora "Maria" está no topo da pilha.

Agora "Ana" está no topo da pilha.

ertholdo@ifsp.edu.br

Referências

- Alfredo Lotar; ASP.NET com C# – Curso Prático. Novatec, 2003.
- Cláudio Vieira Oliveira, Ângela Lühmann e Benedito Petroni; Visual Studio C# – Fundamentos, Programação com ASP.NET, Windows Forms e Web Services. Editora Ciência Moderna, 2015.
- John Sharp; Microsoft Visual C# 2013: Passo a Passo. Bookman, 2014.
- http://www.macoratti.net/12/12/c_col1.htm