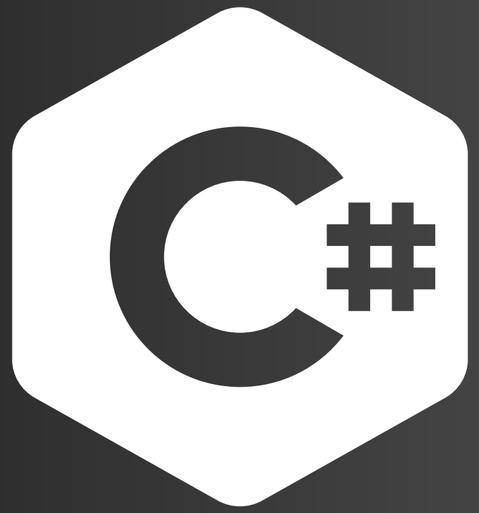


Estruturas de Dados

Prof. Wanderson Timóteo
www.wandersontimoteo.com.br

Wanderson Timóteo

O que vamos aprender

- O que é Estruturas de Dados?
- O que são Coleções em C#?
- Tipos de Coleções em C#:

Wanderson Timóteo

O que vamos aprender

- **O que é Estruturas de Dados?**
- ~~O que são Coleções em C#?~~
- ~~Tipos de Coleções em C#:~~

Estruturas de Dados

O que é Estruturas de Dados?

As estruturas de dados referem-se a várias maneiras de organizar, gerenciar e armazenar dados de forma eficiente para que possam ser acessados e manipulados conforme necessário. Estruturas de dados incluem vetores, matrizes, listas, listas ligadas, árvores, filas, pilhas e grafos, cada uma com suas próprias características de desempenho e uso.

O que vamos aprender

- ~~O que é Estruturas de Dados?~~
- **O que são Coleções em C#?**
- ~~Tipos de Coleções em C#:~~

Wanderson Timóteo

Estruturas de Dados

Coleções em C#

As coleções em C# são implementações específicas para cada tipo de estrutura de dado. Estas coleções permitem armazenar e gerenciar grupos de objetos de forma eficiente e organizada. Elas fazem parte do namespace System.Collections (não-genéricas) e System.Collections.Generic (genéricas) e **são projetadas para manipular conjuntos de dados, oferecendo métodos e propriedades que facilitam operações como adicionar, remover, buscar e ordenar elementos.**

O que vamos aprender

- ~~O que é Estruturas de Dados?~~
- ~~O que são Coleções em C#?~~
- **Tipos de Coleções em C#;**

Wanderson Timóteo

Tipos de Coleções em C#

Implementação de Estruturas de Dados em C#

- Vetor (Array);
- Lista (List);
- Fila (Queue);
- Pilha (Stack);
- Conjunto (Set);
- Graph (Grafo);
- Matriz (Array Multidimensional);
- Lista Ligada (LinkedList);
- Árvore Binária (TreeNode e BinaryTree)
- Dicionário (Dictionary);
- Fila de Prioridade (Priority Queue);
- Hashtable;
- SortedList;
- SortedSet;
- Trie;

Estas são algumas estruturas de dados que você pode usar em C#. Cada uma serve a um propósito específico e pode ser escolhida de acordo com as necessidades de eficiência em seu projeto.

Tipos de Coleções em C#

Implementação de Estruturas de Dados em C#

- **Vetor (Array);**
- ~~Lista (List);~~
- ~~Conjunto (Set);~~
- ~~Fila (Queue);~~
- ~~Pilha (Stack);~~
- ~~Graph (Grafo);~~
- ~~Matriz (Array Multidimensional);~~
- ~~Lista Ligada (LinkedList);~~
- ~~Árvore Binária (TreeNode e BinaryTree)~~
- ~~Dicionário (Dictionary);~~
- ~~Fila de Prioridade (Priority Queue);~~
- ~~Hashtable;~~
- ~~SortedList;~~
- ~~SortedSet;~~
- ~~Trie;~~

Wanderson Timóteo

Estruturas de Dados

Vetor (Array)

Array é uma estrutura de dados que permite armazenar uma **coleção de elementos** sob um único nome, **uma variável**, onde cada elemento é acessado por meio de um **índice único**. Em outras palavras, um array é uma coleção ordenada de elementos que são armazenados de forma **contígua na memória** e cada elemento pode ser identificado por um número inteiro chamado índice.

Estruturas de Dados

Vetor (Array)

O que significa, armazenados de forma contígua na memória?

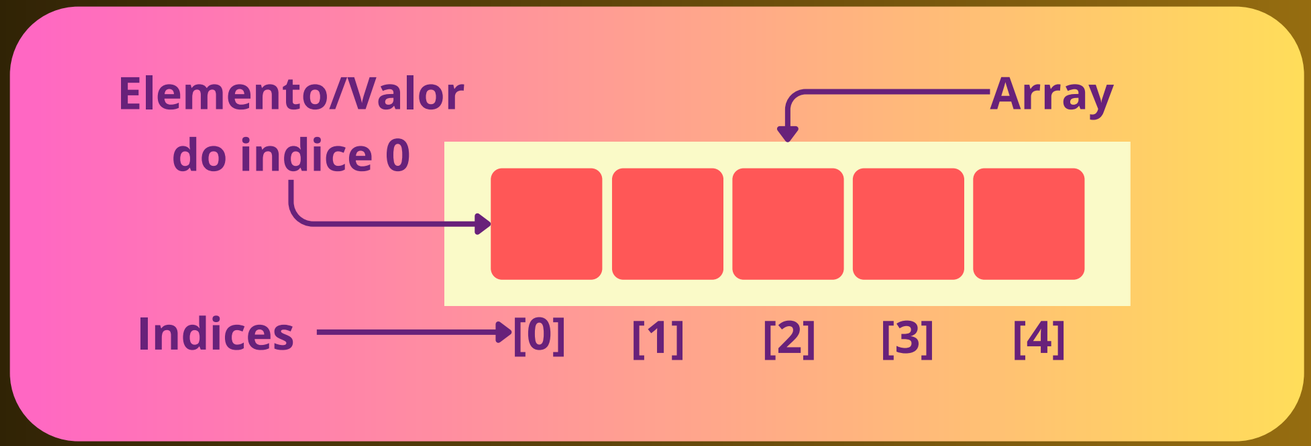
Significa que os elementos de um array são armazenados um após o outro, ocupando posições consecutivas na memória **RAM** do computador. Isso significa que, em um array, cada elemento é adjacente ao próximo na memória, sem lacunas entre eles.

Estruturas de Dados

Vetor (Array)

Essa característica de armazenamento contíguo na memória também significa que os arrays são particularmente eficientes em termos de uso de memória, pois não há espaço desperdiçado entre os elementos.

A figura a baixo mostra a representação de um array com 5 elementos na memória.



Wanderson Timóteo

Estruturas de Dados

Vetor (Array)

Neste exemplo, foi criado um array do tipo inteiro, o seu tamanho com 5 elementos.

```
int[] meuArray = { 1, 2, 3, 4, 5 };
```

Definimos o tamanho do array no momento de sua criação e este não pode ser alterado.

Tipos de Coleções em C#

Implementação de Estruturas de Dados em C#

- ~~Vetor (Array);~~
- **Lista (List);**
- ~~Conjunto (Set);~~
- ~~Fila (Queue);~~
- ~~Pilha (Stack);~~
- ~~Graph (Grafo);~~
- ~~Matriz (Array Multidimensional);~~
- ~~Lista Ligada (LinkedList);~~
- ~~Árvore Binária (TreeNode e BinaryTree)~~
- ~~Dicionário (Dictionary);~~
- ~~Fila de Prioridade (Priority Queue);~~
- ~~Hashtable;~~
- ~~SortedList;~~
- ~~SortedSet;~~
- ~~Trie;~~

Estruturas de Dados

Lista (List)

Em C#, a List é uma coleção genérica que faz parte do namespace System.Collections.Generic e representa uma lista de objetos que pode crescer ou diminuir dinamicamente. Ao contrário dos arrays de tamanho fixo, a List<T> permite adicionar, remover, buscar e ordenar elementos de forma eficiente. Essa flexibilidade faz dela uma das estruturas de dados mais usadas para manipulação de conjuntos de dados em C#.
Para usar uma lista em C#, é necessário importar o namespace System.Collections.Generic.

Wanderson Timóteo

Estruturas de Dados

Lista (List)

Coleção que permite redimensionamento dinâmico, ou seja, adicionar e remover elementos facilmente. Para sua implementação usamos a classe List<T>.

Exemplo:

```
List<string> listaNomes = new List<string> {“Wanderson”, “Jeane”, “Ryan”};
listaNomes.Add(“Maria”);
```

Tipos de Coleções em C#

Implementação de Estruturas de Dados em C#

- ~~Vetor (Array);~~
- ~~Lista (List);~~
- **Conjunto (Set);**
- ~~Fila (Queue);~~
- ~~Pilha (Stack);~~
- ~~Graph (Grafo);~~
- ~~Matriz (Array Multidimensional);~~
- ~~Lista Ligada (LinkedList);~~
- ~~Árvore Binária (TreeNode e BinaryTree)~~
- ~~Dicionário (Dictionary);~~
- ~~Fila de Prioridade (Priority Queue);~~
- ~~Hashtable;~~
- ~~SortedList;~~
- ~~SortedSet;~~
- ~~Trie;~~

Estruturas de Dados

Conjunto (Set)

Um conjunto é uma estrutura de dados que representa uma coleção de valores distintos, sem manter uma ordem específica. No .NET, a estrutura de dados mais comum para conjuntos é o HashSet<T>, que faz parte do namespace System.Collections.Generic.
Como o HashSet não permite valores duplicados e não garante a ordem de inserção, os elementos podem ser exibidos em qualquer ordem.

Estruturas de Dados

Conjunto (Set)

Para implementar um conjunto em C#, usamos HashSet<T> uma classe genérica que implementa várias interfaces de coleção.

```
HashSet<int> conjunto = new HashSet<int> { 1, 2, 3 };
conjunto.Add(4);
```

Wanderson Timóteo

Exercícios

1. Dado um array de inteiros int[] numeros = { 1, 2, 3, 4, 5 };, escreva um código que some todos os elementos do array e exiba o resultado no console.
2. Dada uma lista de strings List<string> nomes = new List<string> { "Jeane", "Wanderson", "Jeane", "Ryan", "Jeane" };, escreva um código para contar quantas vezes o nome "Jeane" aparece na lista.
3. Dada uma lista de inteiros com elementos duplicados List<int> numeros = new List<int> { 1, 2, 2, 3, 4, 4, 5 };, escreva um código para remover os elementos duplicados e exibir a lista sem duplicatas.
4. Dado um conjunto HashSet<string> frutas = new HashSet<string> { "Maçã", "Banana", "Laranja" };, escreva um código que verifique se "Banana" está no conjunto e exiba uma mensagem no console.
5. Crie uma lista vazia de números inteiros List<int> numeros = new List<int>();. Em seguida, adicione os números de 1 a 5 na lista e exiba os elementos no console.

Referências

- https://learn.microsoft.com/pt-br/dotnet/standard/collections/
- https://learn.microsoft.com/pt-br/dotnet/standard/collections/selecting-a-collection-class
- https://learn.microsoft.com/pt-br/dotnet/standard/collections/commonly-used-collection-types
- https://learn.microsoft.com/pt-br/dotnet/standard/collections/when-to-use-generic-collections
- https://learn.microsoft.com/pt-br/dotnet/standard/collections/comparisons-and-sorts-within-collections