Coleções de Dados em C#: Métodos Mais Usados

1 Introdução

Este documento apresenta os métodos mais utilizados nas principais coleções de dados em C#, incluindo Array, Matriz, List, Dictionary, HashSet e Queue. Cada seção descreve os métodos comuns, suas funcionalidades e exemplos práticos.

2 Array

Os arrays em C# são coleções de tamanho fixo que armazenam elementos do mesmo tipo. Eles são indexados a partir de zero.

2.1 Métodos Comuns

- `Length`: Retorna o número total de elementos no array.
- `Clear()`: Define um intervalo de elementos como zero, nulo ou falso.
- `Copy()`: Copia elementos de um array para outro.
- `IndexOf()`: Retorna o índice da primeira ocorrência de um valor.
- `Reverse()`: Inverte a ordem dos elementos.
- `Sort()`: Ordena os elementos em ordem crescente.

2.2 Exemplo

```
int[] numeros = { 3, 1, 4, 1, 5 };
Console.WriteLine(numeros.Length); // Saída: 5
Array.Sort(numeros); // Ordena: { 1, 1, 3, 4, 5 }
Console.WriteLine(Array.IndexOf(numeros, 4)); // Saída: 3
Array.Reverse(numeros); // Inverte: { 5, 4, 3, 1, 1 }
```

3 Matriz (Array Multidimensional)

Matrizes são arrays multidimensionais, comumente usados para representar tabelas ou grades.

3.1 Métodos Comuns

- `GetLength(int dimension)`: Retorna o tamanho de uma dimensão específica.
- `Rank`: Retorna o número de dimensões da matriz.
- `GetUpperBound(int dimension)`: Retorna o índice superior de uma dimensão.

3.2 Exemplo

```
int[,] matriz = { { 1, 2 }, { 3, 4 } };
Console.WriteLine(matriz.GetLength(0)); // Saída: 2 (linhas)
Console.WriteLine(matriz.GetLength(1)); // Saída: 2 (colunas)
Console.WriteLine(matriz.Rank); // Saída: 2 (dimensões)
```

4 List<T>

A classe `List<T>` é uma coleção dinâmica que permite adicionar, remover e modificar elementos.

4.1 Métodos Comuns

- `Add(T item)`: Adiciona um elemento ao final da lista.
- `Remove(T item)`: Remove a primeira ocorrência de um elemento.
- `RemoveAt(int index)`: Remove o elemento no índice especificado.
- `Contains(T item)`: Verifica se um elemento está na lista.
- `Clear()`: Remove todos os elementos.
- `Sort()`: Ordena os elementos.
- `Find(Predicate<T> match)`: Retorna o primeiro elemento que satisfaz um predicado.

4.2 Exemplo

```
List<int> lista = new List<int> { 1, 2, 3 };
lista.Add(4); // Adiciona: { 1, 2, 3, 4 }
lista.Remove(2); // Remove: { 1, 3, 4 }
Console.WriteLine(lista.Contains(3)); // Saída: True
lista.Sort(); // Ordena: { 1, 3, 4 }
int encontrado = lista.Find(x => x > 2); // Saída: 3
```

5 Dictionary<TKey, TValue>

O `Dictionary<TKey, TValue>` armazena pares chave-valor, onde cada chave é única.

5.1 Métodos Comuns

- `Add(TKey key, TValue value)`: Adiciona um par chave-valor.
- `Remove(TKey key)`: Remove o par com a chave especificada.
- `ContainsKey(TKey key)`: Verifica se a chave existe.
- `TryGetValue(TKey key, out TValue value)`: Tenta obter o valor associado a uma chave.
- `Clear()`: Remove todos os pares.
- `Keys`: Retorna uma coleção de chaves.
- `Values`: Retorna uma coleção de valores.

5.2 Exemplo

```
Dictionary<string, int> dict = new Dictionary<string, int>();
dict.Add("um", 1); // Adiciona: { "um", 1 }
dict.Add("dois", 2);
Console.WriteLine(dict.ContainsKey("um")); // Saída: True
int valor;
dict.TryGetValue("dois", out valor); // Saída: 2
dict.Remove("um"); // Remove: { "dois", 2 }
```

6 HashSet<T>

O `HashSet<T>` é uma coleção que armazena elementos únicos, sem ordem específica.

6.1 Métodos Comuns

- `Add(T item)`: Adiciona um elemento, se não existir.
- `Remove(T item)`: Remove o elemento especificado.
- `Contains(T item)`: Verifica se o elemento existe.
- `Clear()`: Remove todos os elementos.
- `UnionWith(IEnumerable<T> other)`: Adiciona elementos de outra coleção.

6.2 Exemplo

```
HashSet<int> set = new HashSet<int> { 1, 2, 3 };
set.Add(3); // Ignorado (já existe)
set.Add(4); // Adiciona: { 1, 2, 3, 4 }
Console.WriteLine(set.Contains(2)); // Saída: True
set.Remove(1); // Remove: { 2, 3, 4 }
```

7 Queue<T>

A `Queue<T>` é uma coleção do tipo FIFO (primeiro a entrar, primeiro a sair).

7.1 Métodos Comuns

- `Enqueue(T item)`: Adiciona um elemento ao final da fila.
- `Dequeue()`: Remove e retorna o elemento no início da fila.
- `Peek()`: Retorna o elemento no início sem removê-lo.
- `Clear()`: Remove todos os elementos.
- `Count`: Retorna o número de elementos na fila.

7.2 Exemplo

```
Queue<string> fila = new Queue<string>();
fila.Enqueue("A"); // Adiciona: { "A" }
fila.Enqueue("B"); // Adiciona: { "A", "B" }
Console.WriteLine(fila.Dequeue()); // Saída: A
Console.WriteLine(fila.Peek()); // Saída: B
Console.WriteLine(fila.Count); // Saída: 1
```