C# Array Functions Examples (Single, Multi-Dimensional, Jagged)

# 1. Array.Length

Explanation:

Returns the number of elements in the array.

Code:

// Single-dimensional  
int[] temperatures = { 23, 19, 31, 27, 30, 25 };  
Console.WriteLine(temperatures.Length); // Output: 6  
  
// Multi-dimensional  
int[,] matrix = { {1, 2}, {3, 4} };  
Console.WriteLine(matrix.Length); // Output: 4  
  
// Jagged array  
int[][] jagged = { new int[] {1, 2}, new int[] {3, 4, 5} };  
Console.WriteLine(jagged.Length); // Output: 2 (outer array)

# 2. Array.Rank

Explanation:

Returns the number of dimensions in the array.

Code:

// Single-dimensional  
int[] numbers = { 10, 20, 30 };  
Console.WriteLine(numbers.Rank); // Output: 1  
  
// Multi-dimensional  
int[,] grid = new int[3, 4];  
Console.WriteLine(grid.Rank); // Output: 2  
  
// Jagged array  
int[][] jagged = new int[3][];  
Console.WriteLine(jagged.Rank); // Output: 1 (outer array is 1D)

# 3. Array.IndexOf

Explanation:

Finds the index of a value in a 1D array. Doesn't work with 2D arrays.

Code:

// Single-dimensional  
string[] fruits = { "Apple", "Banana", "Cherry" };  
int index = Array.IndexOf(fruits, "Banana");  
Console.WriteLine(index); // Output: 1  
  
// Jagged array (match by reference)  
int[] row1 = { 1, 2, 3 };  
int[] row2 = { 4, 5, 6 };  
int[][] jagged = { row1, row2 };  
int idx = Array.IndexOf(jagged, row2);  
Console.WriteLine(idx); // Output: 1  
  
// 2D array - Invalid  
int[,] matrix = { { 1, 2 }, { 3, 4 } };  
// Array.IndexOf(matrix, 3); // ❌ Compile-time error