Array in C#

An array in C# is a data structure that stores a fixed-size sequential collection of elements of the same type. Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

Key Features of Arrays:

1. **Fixed Size**: The size of an array is defined when it is created and cannot be changed.
2. **Same Type**: All elements in an array must be of the same type (e.g., all integers, all strings, etc.).
3. **Indexing**: Arrays are zero-indexed, meaning the first element is at index 0, the second at index 1, and so on.
4. **Efficient Access**: Arrays provide fast access to elements using their index.

Declaring and Initializing Arrays

1. Declaring an Array

You declare an array by specifying the data type of its elements and the number of elements it can hold.

```
int[] numbers; // Declares an array of integers
```

2. Initializing an Array

After declaring an array, you can initialize it by specifying the size of the array or by assigning values directly.

Example 1: Initializing with Size

```
int[] numbers = new int[5]; // Initializes an array with 5 elements, all set to 0 by default
```

Example 2: Initializing with Values

```
int[] numbers = new int[] { 1, 20, 13, 4, 5 }; // Initializes an array with the values 1, 2, 3, 4, 5
```

You can also omit the new int[] part when providing the values directly:

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

Accessing Array Elements

You can access elements in an array using their index.

Example:

```
// create array for tank temperature ( single line comment )
/*
Ignored block ( multiline comment )
*/
int[] TankTempDegrees = { 1, 2, 3, 4, 5 };
Console.WriteLine(TankTempDegrees [0]); // Outputs 1 (first element)
Console.WriteLine(TankTempDegrees [4]); // Outputs 5 (fifth element)
```

Modifying Elements:

```
numbers[2];
numbers[2] = 10; // Sets the third element to 10
Console.WriteLine(numbers[2]); // Outputs 10
```

Iterating Over Arrays

You can use loops to iterate over the elements of an array.

Using for Loop:

```
for (int i = 0; i < numbers.Length; i++)
{
   Console.WriteLine(numbers[i]);
}
```

Using foreach Loop:

```
foreach (int number in numbers)
{
   Console.WriteLine(number);
}
```

Multi-Dimensional Arrays

C# supports multi-dimensional arrays, which can be thought of as arrays of arrays. The most common multi-dimensional array is the two-dimensional array (matrix).

Declaring a Two-Dimensional Array:

```
int[,] matrix = new int[3, 3]; // Declares a 3x3 matrix (3 rows, 3 columns)
```

Initializing a Two-Dimensional Array:

```
int[,] matrix = {
  { 1, 2, 3 },
  { 4, 5, 6 },
  { 7, 8, 9 }
};
```

Accessing Elements:

```
Console.WriteLine(matrix[0, 0]); // Outputs 1 (element in the first row, first column)
Console.WriteLine(matrix[2, 2]); // Outputs 9 (element in the third row, third column)
```

Jagged Arrays

A jagged array is an array of arrays, where each "inner" array can have different lengths.

Declaring a Jagged Array:

```
int[][] jaggedArray = new int[3][]; // Declares a jagged array with 3 inner arrays
```

Initializing a Jagged Array:

```
jaggedArray[0] = new int[] { 1, 2, 3 };
jaggedArray[1] = new int[] { 4, 5 };
jaggedArray[2] = new int[] { 6, 7, 8, 9 };
```

Accessing Elements in a Jagged Array:

```
Console.WriteLine(jaggedArray[0][0]); // Outputs 1 (first element of the first array)
Console.WriteLine(jaggedArray[1][1]); // Outputs 5 (second element of the second array)
```

Array Methods and Properties

C# provides several useful methods and properties for working with arrays:

- **Length**: Returns the number of elements in the array.

```
int[] numbers = { 1, 2, 3, 4, 5 };
Console.WriteLine(numbers.Length); // Outputs 5
```

- **Array.Sort()**: Sorts the elements of the array.

```
int[] numbers = { 5, 1, 4, 2, 3 };
Array.Sort(numbers);
Console.WriteLine(string.Join(", ", numbers)); // Outputs 1, 2, 3, 4, 5
```

- **Array.Reverse()**: Reverses the order of the elements in the array.

```
int[] numbers = { 1, 2, 3, 4, 5 };
Array.Reverse(numbers);
Console.WriteLine(string.Join(", ", numbers)); // Outputs 5, 4, 3, 2, 1
```

- **Array.IndexOf()**: Returns the index of the first occurrence of a value.

```
int[] numbers = { 1, 2, 3, 4, 5 };
int index = Array.IndexOf(numbers, 3);
Console.WriteLine(index); // Outputs 2
```