Working with Arrays in C#

1. Introduction to Arrays

- An **Array** is a collection of **similar data types** stored in **contiguous memory locations**.
- Arrays allow storing multiple values under a single variable name.
- Useful when working with large data sets like numbers, strings, or objects.

2. Array Characteristics

- Fixed size (declared during initialization).
- Index-based (0-based indexing).
- Homogeneous (all elements must be of the same data type).

3. Declaring and Initializing Arrays

Syntax:

```
datatype[] arrayName = new datatype[size];
```

Example:

```
int[] numbers = new int[5];
```

Initialization at Declaration:

```
int[] numbers = { 10, 20, 30, 40, 50 };
```

4. Accessing Array Elements

```
Console.WriteLine(numbers[0]);  // Access first element
numbers[1] = 25;                // Modify second element
```

Index	0	1	2	3	4
Value	10	25	30	40	50

5. Traversing Arrays (Loops with Arrays)

Using for loop:

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

Using foreach loop:

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

---

6. Array Properties and Methods

Property/Method	Description
`Length`	Returns the total number of elements
`Rank`	Returns the number of dimensions
`Array.Sort(arr)`	Sorts the array elements
`Array.Reverse()`	Reverses the array elements

---

7. Types of Arrays in C#

1. Single-Dimensional Array

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

2. Multi-Dimensional Array (Matrix)

```
int[,] matrix = { {1, 2}, {3, 4}, {5, 6} };
Console.WriteLine(matrix[1, 1]); // Output: 4
```

3. Jagged Array (Array of Arrays)

```
int[][] jagged = new int[2][];
jagged[0] = new int[] {1, 2, 3};
jagged[1] = new int[] {4, 5};
Console.WriteLine(jagged[0][2]);  // Output: 3
```

---

8. Common Array Operations

Sum of Array Elements

```
int sum = 0;
foreach (int num in numbers)
{
    sum += num;
}
Console.WriteLine("Sum: " + sum);
```

Finding Maximum Element

```
int max = numbers[0];
for (int i = 1; i < numbers.Length; i++)
{
    if (numbers[i] > max)
        max = numbers[i];
}
Console.WriteLine("Max: " + max);
```

---

9. Real-life Example - Storing Student Marks

```
int[] marks = new int[5];
Console.WriteLine("Enter marks of 5 subjects:");
for (int i = 0; i < 5; i++)
{
    marks[i] = Convert.ToInt32(Console.ReadLine());
}

int total = 0;
foreach (int mark in marks)
{
    total += mark;
}
Console.WriteLine("Total Marks: " + total);
Console.WriteLine("Average Marks: " + (total / 5));
```

10. Array Limitations

- Fixed size (cannot grow/shrink after declaration).
- Only stores one data type.
- Does not support dynamic resizing.

> 🔎 For dynamic collections, C# provides **List** from **System.Collections.Generic**.

11. Key Points Recap

- Arrays store multiple values of the same type.
- Index starts from `0`.
- Use `for` or `foreach` loops for traversing.
- Supports single, multi-dimensional, and jagged arrays.

12. Homework / Practice Problems

1. Write a program to read 10 integers in an array and print them in reverse order.
2. Create a 2D array and find the sum of all elements.
3. Store 5 names in a string array and search for a name entered by the user.
4. Implement a jagged array to store different number of elements in each row.