**Part01**

**Question: What is the default value assigned to array elements in C#?**

Default value is null.

**Question: What is the difference between Array.Clone() and Array.Copy()?**

* **Array.Clone()**

Shallow Copy: This method creates a shallow copy of the array, meaning it copies the array structure and references but does not copy the objects referenced by the array.

* **Array.Copy()**

Deep Copy: It copies all the elements from the source array to the destination array. This method can be used to copy a part or the entire array.

**Question: What is the difference between GetLength() and Length for multidimensional**

**arrays?**

* GetLength(dimension) is used to find the size of a specific dimension in a multidimensional array.

**EX:** int[,] multiArray = new int[4, 5];

int rows = multiArray.GetLength(0); // Returns 4 (number of rows)

int columns = multiArray.GetLength(1); // Returns 5 (number of columns)

* Length is used to find the total number of elements in the array, regardless of its dimensions.

**EX:** int[,] multiArray = new int[4, 5];

int totalElements = multiArray.Length; // Returns 20 (4\*5)

**Question: What is the difference between Array.Copy() and**

**Array.ConstrainedCopy()?**

* Array.Copy()

This method is commonly used to copy elements from one array to another , but It doesn't enforce strict type compatibility between the source and destination arrays. If there is an issue with the copy operation, it may not always provide detailed error information.

* Array.ConstrainedCopy() : commonly used to copy elements from one array to another and enforces stricter type checking and ensures that the source and destination arrays have compatible element types

**Question: Why is foreach preferred for read-only operations on arrays?**

* makes the code more readable and concise by abstracting the details of iteration.
* does not allow modification of the elements during iteration. This immutability ensures that the data remains unchanged, making it ideal for read-only operations.
* **but it does come with a few limitations that make it suitable primarily for read-only operations:**
* The iteration variable in a foreach loop is read-only, meaning you cannot modify the elements of the array or collection directly through this variable.
* does not provide access to the index of the current element.

**Question: Why is input validation important when working with user inputs?**

* + Prevents system crashes due to unexpected data.
  + Ensures the application handles inputs gracefully.
  + Prevents errors and corrupt data.
  + Prevents injection attacks (e.g., SQL injection, XSS).
  + Minimizes exploits and security breaches.

**Question: How can you format the output of a 2D array for better readability?**

**1-Tabular Format**:

* Use nested loops to iterate through the rows and columns.
* Print elements with consistent spacing.

**int[,] array = {**

**{1, 2, 3},**

**{4, 5, 6},**

**{7, 8, 9}**

**};**

**for (int i = 0; i < array.GetLength(0); i++)**

**{**

**for (int j = 0; j < array.GetLength(1); j++)**

**{**

**Console.Write(array[i, j].ToString().PadRight(4));**

**}**

**Console.WriteLine();**

**}**

**2- Using String Formatting:**

* Use String.Format or Console.Write with format specifiers to align columns

**int[,] array = {**

**{1, 20, 300},**

**{4000, 50, 6},**

**{7, 800, 90}**

**};**

**for (int i = 0; i < array.GetLength(0); i++)**

**{**

**for (int j = 0; j < array.GetLength(1); j++)**

**{**

**Console.Write("{0,5}", array[i, j]);**

**}**

**Console.WriteLine();**

**}**

**3- Using Libraries:**

* Use libraries like System.Data or third-party libraries like ConsoleTables for more advanced formatting**.**

**using ConsoleTables;**

**var table = new ConsoleTable("Col1", "Col2", "Col3");**

**int[,] array = {**

**{1, 2, 3},**

**{4, 5, 6},**

**{7, 8, 9}**

**};**

**for (int i = 0; i < array.GetLength(0); i++)**

**{**

**table.AddRow(array[i, 0], array[i, 1], array[i, 2]);**

**}**

**table.Write();**

**Question: When should you prefer a switch statement over if-else?**

* When you have multiple conditions based on the same variable or expression, a switch statement can make the code more readable and easier to manage.
* Especially when the values are well-defined and limited. It helps improve code readability, maintainability, and can offer performance benefits.

**Question: What is the time complexity of Array.Sort()?**

* **The Worst Case**: O(nlog n)

**Question: Which loop (for or foreach) is more efficient for calculating the sum of an**

**array, and why?**

**·** Control: The for loop gives you direct control over the index, allowing you to modify it within the loop body if needed.

· Simplicity: The foreach loop is more concise and easier to read, especially for simply iterating over each element without needing the index.

· Performance:

In modern C#, the performance difference between for and foreach is minimal due to compiler optimizations.

The for loop might be slightly faster in some cases because it avoids the overhead of the IEnumerator interface used by foreach. However, this difference is often negligible and unlikely to impact most applications.

Both are efficient but we can say for loop is better as it might be slightly faster.

Part02

1. LinkedIn article about loops statements in Csharp

3-What happens if the user enters a value outside the range of 1 to 7?

If the user enters 8, for example, the program will output:8

Even though 8 is not a valid WeekDays value, the enum will still interpret it as 8, which can be misleading. This happens because enums in C# are essentially named constants for integer values, and casting an integer to an enum type will not throw an error, even if the integer is outside the defined range of the enum.