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

In C#, the default values depend on the type of the array:

- For int, float, double: the default is 0.
- For bool: the default is false.
- For char: the default is '\0' (null character).
- For string and reference types: the default is null.
- For int : the default is 0.

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

Array.Clone()

Creates a shallow copy of the entire array.

A new array of the same type and length

Changes in arr2 don't affect arr1 (because it's a new array),

but if the array contains reference types (like objects), both arrays still point to the same objects.

Array.Copy()

Copies elements from one array to another, starting at specified indexes. Nothing (it's a void method).

What is the difference between GetLength() and Length for multi dimensional arrays?

Length

• Returns: The total number of elements in the entire array (all dimensions).

GetLength(dimension)

- Returns: The number of elements in a specific dimension.
- Parameter: Takes an int index (starting from 0 for the first dimension).

What is the difference between Array.Copy() and Array.ConstrainedCopy()?

Array.Copy()

• Purpose:

Copies elements from one array to another.

- Behavior:
 - Performs a shallow copy.
 - Fast and commonly used.
 - Less strict with type safety and exception rollback.

Array.ConstrainedCopy()

- Purpose:
 - Like Array.Copy(), but with additional safety for copying across AppDomains and ensuring atomicity (no partial copy).
- Behavior:
 - o Performs extra checks to ensure type safety.
 - o If any error occurs, no data is copied at all (rollback).
 - Slower due to these checks.

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

- foreach makes the code **cleaner** and easier to understand.
- You don't have to deal with indexes manually.
- foreach is **read-only** by design.
- You can't modify the array elements inside a foreach loop (directly).
- This protects the array from accidental changes.

Why is input validation important when working with user inputs?

Prevents Errors and Crashes

Enhances Security

input validation helps prevent security vulnerabilities like:

- SQL Injection
- Cross-site Scripting (XSS)
- Command Injection

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

- PadLeft (4) ensures each number takes at least 4 characters of space, so columns are aligned.
- GetLength(0) gives the number of rows.
- GetLength (1) gives the number of columns.
- Console.WriteLine() moves to the next line after each row.

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

- switch is more readable and cleaner than many if-else blocks.
- switch helps prevent missing conditions.
- The default block can catch any unexpected values.
- switch can be faster than if-else when compiled (especially with many cases).

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

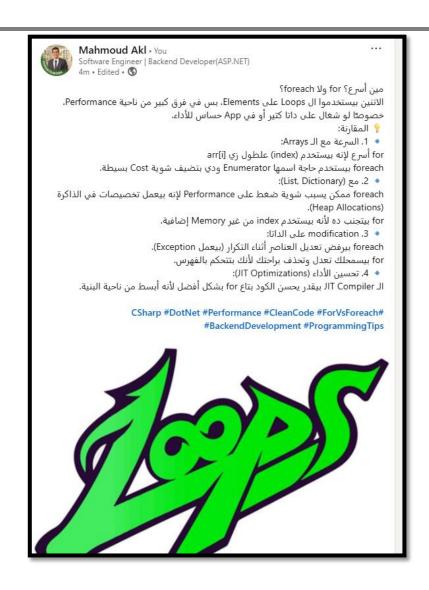
Case	Time Complexity
Average case (most of the time)	O(n log n)
Best case (data is nearly sorted)	O(n log n)
Worst case (data is very unsorted)	O(n log n)

Which loop (for or foreach) is more efficient for calculating the sum of an array, and why?

for loop is slightly more efficient in some scenarios:

Why?

- for uses an **index**, so the compiler can optimize it better.
- foreach uses an **enumerator** internally, which adds a tiny overhead (like creating an object that keeps track of the position).
- This difference is **very small**, and usually **not noticeable** for small or medium arrays.



What's the default size of the stack and heap, and what are the considerations?

☆ Stack:

- Default Size in C# (.NET):
 - o Around 1 MB (1 megabyte) per thread.
- Important Notes:
 - o Stack size is **fixed** when the thread is created.
 - o If the stack usage exceeds its limit \rightarrow you get a **StackOverflowException**.
 - Stored in the stack:
 - Local variables.
 - Method calls (call stack).
 - Value types (unless boxed).

☆ Heap:

- Default Size:
 - o No fixed size it starts small and grows as needed.
 - o On modern systems, it can grow up to **several gigabytes**, depending on available memory.
- Important Notes:
 - o The heap stores:
 - Reference types (like objects, arrays, strings).
 - Managed by the Garbage Collector (GC) in .NET.
 - o Performance can be affected by:
 - Too many object allocations.
 - Frequent garbage collection cycles.

what is time complexity?

Time complexity is a way to describe how the runtime of an algorithm grows as the size of the input (n) increases.

It gives you a **mathematical measure** of the **efficiency** of an algorithm — especially how it performs as the input becomes very large.

Because two different algorithms might do the same task, but one is way faster for big inputs. Time complexity helps us:

- Compare algorithms
- Predict performance
- Optimize code

Q Big-O Notation:

We use **Big-O** notation to express time complexity.

It describes the **upper bound** of time taken by an algorithm — in the worst-case scenario.