TASK
DAY04

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

n C#, the default value of array elements depends on the data type.

Default values in C# arrays:

-int, float, double, long, short → `0`
-bool → `false`
-char → `'\0'` (null character)
-string → `null`
-object / reference types → `null`

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

`Array.Clone()`

-Creates a new array with the same elements as the original array.
-Returns an object, so it must be type-casted.
-Performs a shallow copy.
-Simple and quick when you want a full copy of the array.

`Array.Copy()`

-Copies elements from a source array to a destination array.
-Allows partial copying and control over starting index and length.
-The destination array must be created first.
-Also performs a shallow copy.

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

`Length`

-Returns the **total number of elements** in a multi-dimensional array.
-It does **not** distinguish between rows and columns.

`GetLength(dimension)`

-Returns the size of a specific dimension.
-`dimension = 0` → number of rows
-`dimension = 1` → number of columns

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

`Array.Copy()`

-Copies elements from a source array to a destination array.
-Does **not guarantee atomicity** (if an error happens during copying, the destination array may be partially copied).
-Faster and commonly used.
-Used when you are sure the copy operation is safe.

`Array.ConstrainedCopy()`

-Copies elements atomically (all or nothing).
-If an exception occurs, no changes are made to the destination array.
-Safer, but slightly slower.

-Mainly used in reliability-critical code.

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

`foreach` is preferred for read-only operations on arrays because:

-**Safer**: You cannot accidentally modify the array elements directly.

-**Simpler syntax**: No index handling, so fewer errors.
-**More readable**: Clear intent that you are only reading values.
-**No risk of IndexOutOfRangeException**.

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

Input validation is important to ensure that user inputs are correct, safe, and within expected limits, preventing runtime errors, logical errors, and security issues.

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

You can format a 2D array for better readability by using tabs (`\t`) or string padding (`PadLeft`/`PadRight`) between elements and nested loops to print each row on a new line. This aligns the columns neatly, making the array appear as a clear matrix.

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

ou should prefer a switch statement over if-else when:

1. Multiple discrete cases – you are comparing a single variable against many constant values (e.g., numbers, characters, or enums).

2. Readability – a switch is cleaner and easier to read than a long chain of if-else statements.

3. Maintenance – easier to add or remove cases without affecting other logic.

4. Performance (sometimes) – for some compilers, switch statements can be faster than multiple if-else checks for many cases.

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

Both `for` and `foreach` loops can be used to calculate the sum of an array, but `for` is generally slightly more efficient for arrays.

Reason:

1. **`for` loop**
   -Accesses array elements directly by index: `arr[i]`
   -No extra overhead, works directly on the array memory

2. **`foreach` loop**
   -Uses an enumerator internally (even for arrays)
   -Slight overhead due to enumerator structure, though usually negligible

Define an enum called DayOfWeek with values: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

```
enum DayOfWeek

{

    Monday,

    Tuesday,

    Wednesday,

    Thursday,

    Friday,

    Saturday,

    Sunday

}
```

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

If the user enters a number outside 1–7, the program considers it invalid and displays an error message. No enum conversion occurs, and the program prevents runtime errors.

Bonus :

Stack

-Purpose: Stores value types (int, double, struct) and method call frames (local variables, function parameters, return addresses).
-Default size:
Windows: typically 1 MB per thread (can vary by OS and configuration)
Linux: usually 8 MB per thread

Characteristics:
-Fast access (LIFO – Last In First Out)
-Memory is automatically managed (deallocated when function ends)
-Limited size → stack overflow if too much memory is used (e.g., deep recursion or large local arrays)

Considerations

1. Keep stack allocations small (avoid large arrays or structures)
2. Recursive calls can easily cause stack overflow
3. Thread stack size can sometimes be configured at thread creation

---

Heap

Purpose: Stores reference types (objects, class instances, arrays)
Default size:
-Managed by .NET CLR garbage collector
-Starts small and grows dynamically up to system memory limits
-Exact size depends on OS, CLR, and process architecture (32-bit vs 64-bit)

Characteristics:
-Slower access than stack (requires pointer dereferencing)
-Memory must be managed (garbage collector handles cleanup)

-Can hold large objects that cannot fit on the stack

Considerations

1. Heap memory is limited by system RAM, so excessive allocation can lead to memory pressure

2. Objects should be released as soon as they are no longer needed to avoid GC overhead

3. Frequent allocation and deallocation can trigger garbage collection, affecting performance

What is time complexity?

Time Complexity is a concept in computer science that describes how the runtime of an algorithm grows as the input size increases.

Definition:

> Time complexity measures the number of basic operations an algorithm performs as a function of the input size n. It helps estimate how fast or slow an algorithm is.