Core C# Programming	
Arrays in C#	Array : a collection of the same data type , stored in contiguous memory location and can be accessed via zero based index. Why we need array when we have so many types of variable in C# ? int [] marks = new int [100]; **We need arrays to simplify data storage, processing, and scalability when dealing with multiple related values.** What are the features of Arrays/ in what scenarios we should use them ? Limitation of using arrays in C#? types of arrays in C#
	Single array
	Multi-dimension
	Jagged array
Looping Construct	Loops and Statement
	IF statement
	For loop
	Foreach loop
	Switch statement
	While loop
	Do-while loop

Access modifiers in C#	Access modifiers
	Private
	Public
	Protected
	Internal
	Protected internal
Methods in C#	Methods
	Void method
	With parameter
	Return type method
	Static method
	Instance method
	Namespaces

	Unit testing

today's case study :

User Story 1: Version Control Adoption : CI/CD operations via Github

As a software development team
I want a version control system to track and manage code changes
So that multiple developers can work in parallel without conflicts and we can safely revert to stable versions when needed.

Acceptance Criteria

- All code changes are committed to a central repository
- Each change is traceable (who, what, when)
- Developers can create branches for features and fixes
- Code can be merged and reviewed before release
- Remote repositories enable team-wide collaboration

User Story: Enabling Faster, Safer Software Delivery

As a business leader,
I want development teams to collaborate efficiently and release software updates reliably,
so that the organization can respond faster to market needs while minimizing operational risk.

The Challenge

As teams grow, managing frequent changes becomes complex.
Without structured control and automation:

- Errors go unnoticed
- Releases are delayed
- Accountability becomes unclear
- Customer impact increases

Solution 👍

-Version control to maintain a single , trusted source for all changes.
-Automated delivery pipeline (CI/CD) to validate and deploy updates consistently.

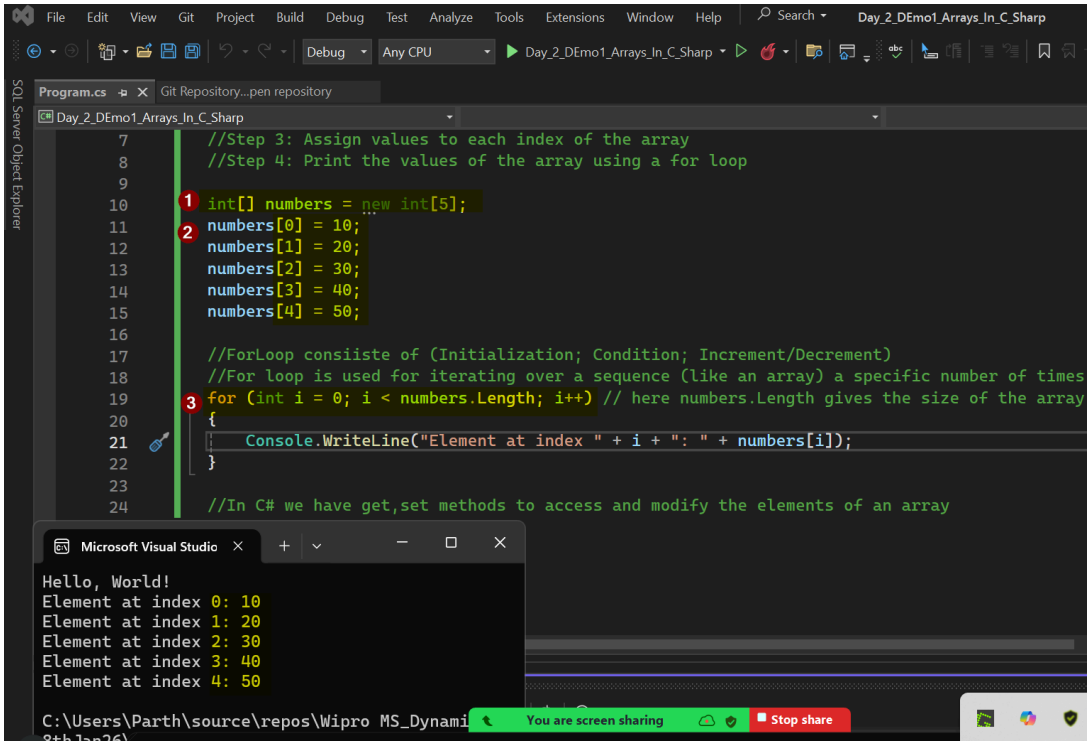
this ensures 👍

- Every change is tracked and auditable.
- Quality checks happens automatically
- deployments are predictable and repeatable.

What is the business outcome of above system adoption :

- Reduced release failure.
- faster time to market.
- Improved product stability
- Higher team productivity
- Better customer trust.

//Steps for creating Arrays in C#
//Step 1: Declare the array of type int
//Step 2: Initialize the array with size 5
//Step 3: Assign values to each index of the array
//Step 4: Print the values of the array using a for loop



//Case study : Using Arrays to Manage Student Grades
//if i want to store marks of students in different subjects: "Subject wise marks for each student"
//Step 1: Declare a 2D array to store marks of 3 students in 4 subjects
//Step 2: Initialize the array with sample marks
//Step 3: Calculate and print the average marks for each student
//Step 4: Calculate and print the average marks for each subject
//Step 5: Find and print the highest and lowest marks in the class
// Syntax for declaraing 2D array as per above sceanario is
// dataType[,] arrayName = new dataType[rows, columns];

// ther is a differenec between storing marks and storing subject wise marks for each student
//In first case each row represents a student and each column represents a subject
//In second case each row represents a subject and each column represents a student

---

A mid-sized **e-commerce platform** is developing an **Order Management System (OMS)** using C#.

The system must:

- Store customer orders
- Add and remove orders dynamically
- Search, sort, and update orders in real time
- Scale during seasonal sales (Flash Sales, Festivals)

order[] orders = new orders[100]; // implementing an array

- **it can take only 100 orders**
- **it can access stored data sequentially.**

In C# we have collections

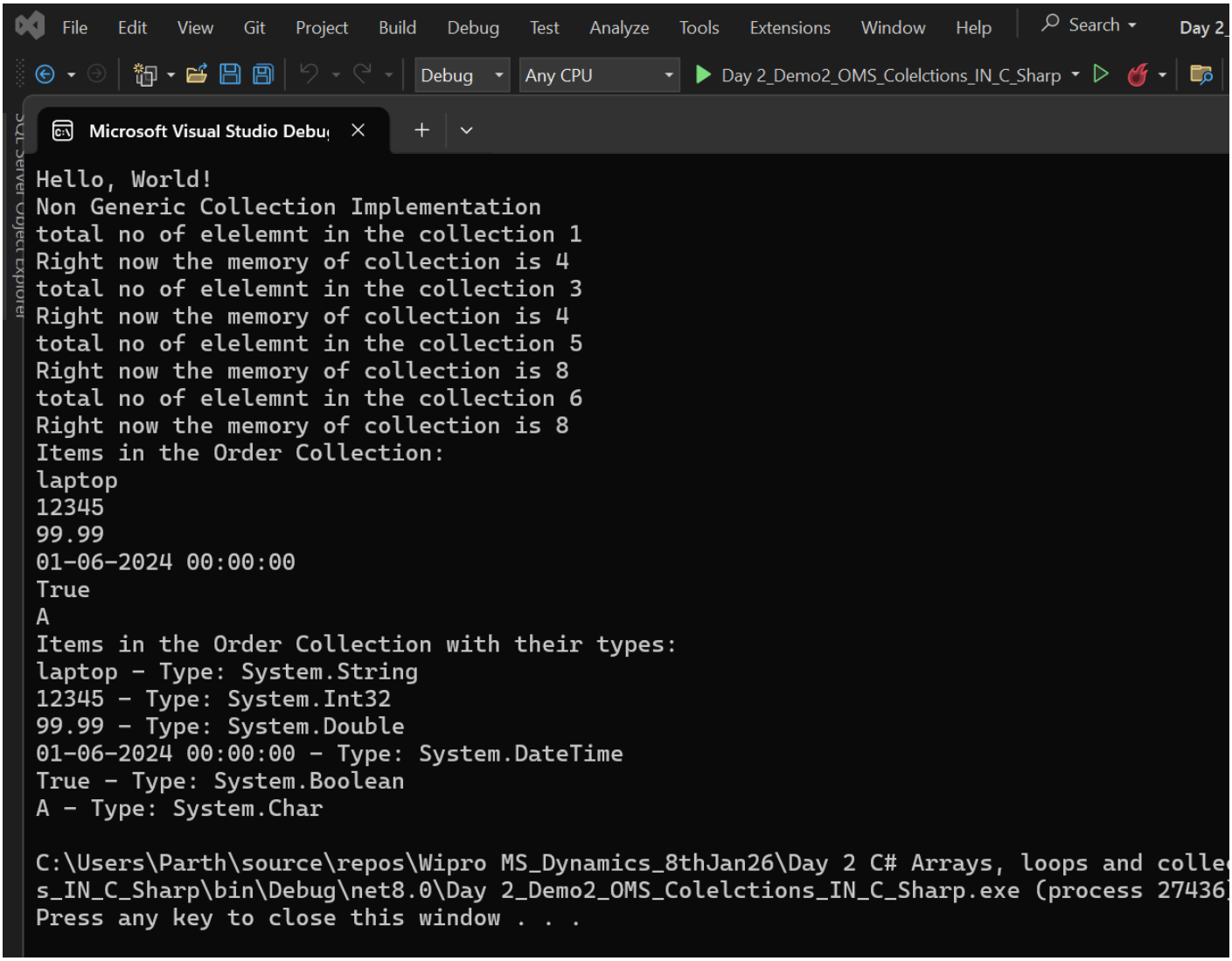
Array Limitation	Collection Solution
Fixed size	Dynamic resizing
Manual insert/delete	`Add()`, `Remove()`
Custom search logic	`Find()`, `Any()`
Sorting complexity	`Sort()`
Memory wastage	Auto capacity management (Common language runtime CLR)

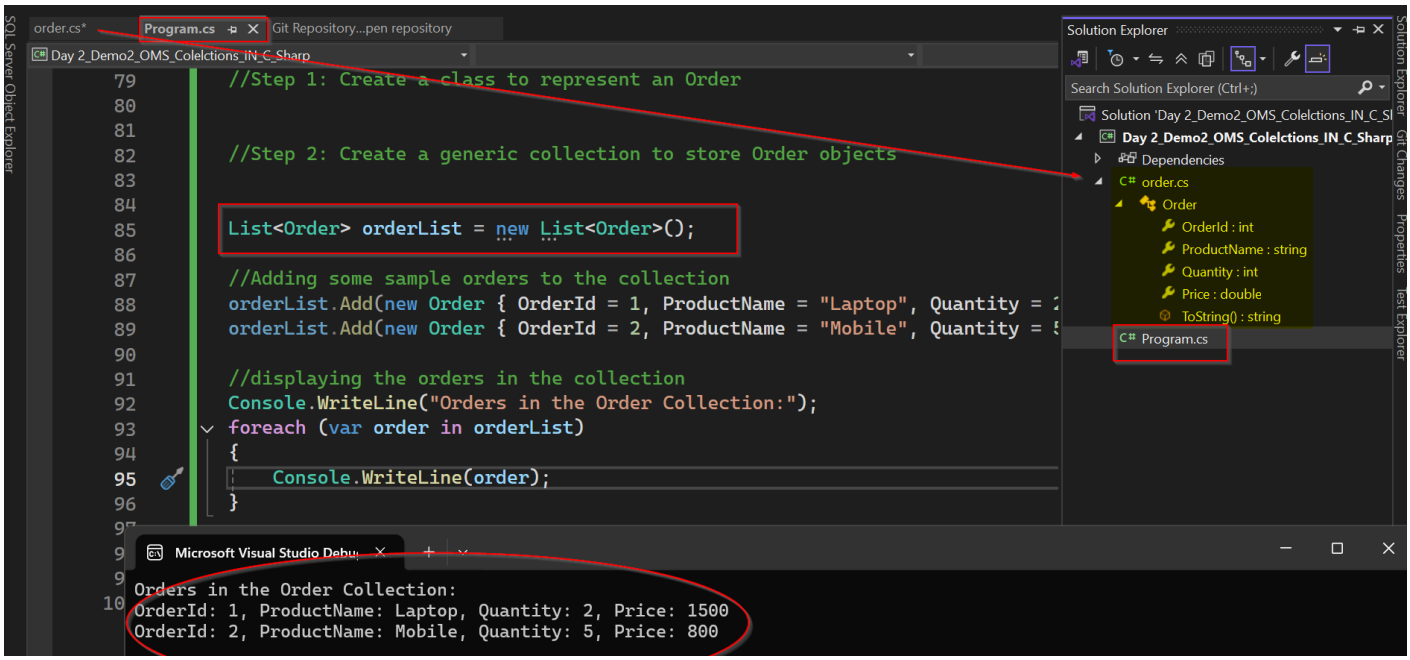
type of collection in C# 👍

1) Generic Collection : List
2) Non generic Collection

Non generic Collection (system.collections)	Generic Collection (system.collections.generic)
Stores data as object .	Stores data as specific type <T>
boxing/unboxing is required for value types	it is not required
performance is slower	faster in performance

compile time errors are not detected	**they are detected**
higher memory consumption	**low memory consumption**
Arraylist , hashtable, Stack, queue, Dictionary(K,V)	**List<t>, hashtable<t>, Stack<t>, queue<t>, Dictionary(tkey>, tvalue>)**





Case Study on types of collections :

A **banking application** needs a **fast lookup mechanism** to fetch **customer profiles** using a **Customer ID**.

System Requirements

- Store thousands of customer records
- Retrieve customer details in real time(faster)
- Ensure data accuracy and performance(need to consider generic Collection)
- Support future scalability(No way arrays can be considered, we have to think in terms of collections only)

Non generic collection hashtable	generic collection Dictionary(tkey, tvalue) //To over overcome above issue we can use

Hashtable customer = new Hashtable();
customer.Add("C001", "John Doe");
customer.Add("C002", "Jane Smith");
customer.Add("C003", "Alice Johnson");
customer.Add("c004", 45000);
	generic dictionary
	Dictionary<string, string> customerGeneric = new Dictionary<string, string>();
	customerGeneric.Add("C001", "John Doe");
	customerGeneric.Add("C002", "Jane Smith");
	customerGeneric.Add("C003", "Alice Johnson");
	customerGeneric.Add("c004", 45000.ToString());//this line will cause compile time error

• No type safety : it is allowing mixed types/value types • run time errors • overhead of performance • poor maintainability	• type safety • no run time error • no overhead • good maintainability

case study based on different types of constructs in C#:

A **Smart Parking System** is developed to manage vehicle entry, parking allocation, billing, and exit processing.

The system must:

- Track parking slots
- Handle vehicle types
- Continuously monitor availability
- Process user choices until exit

1. Loop : for initialized parking slots.
2. for each loop can be used for Displaying parking status
3. Switch : vehicle type handling (car, bike, truck, others)
4. While :
5. Do While

below steps can be used for above ase study implementation 👍

Step 1: Initialize parking with the help of for loop

Step 2: parking vehicles

Step 3: exit vehicles

Step 4: calculate charges based on vehicle type

step 5: Keeping the application running until the user exits.

C# Concept	Used For	Reason
`for`	Slot initialization	Fixed iteration
`foreach`	Display slot status	Safe traversal
`switch`	Vehicle type logic	Multiple conditions
`while`	Slot search	Condition-based
`do-while`	Menu loop	At least once execution

Why we need function in C#? :

1. Reusability
2. Better readability
3. Easier to maintain
4. Modular code

A school needs a small program to:

- Calculate total marks
- Calculate average marks
- Determine pass or fail

Step 1: Function to calculate total marks : should accept parameters and returns a value

Step 2: call function inside main