What Is "Managed Code"?

Managed code is code that runs **under the control of a managed runtime environment**, such as the **.NET CLR**. This environment provides services that automate many aspects of program execution.

Key Features of Managed Code in C#

1. Memory Management

- Automatic memory allocation and **garbage collection**.
- No need to manually allocate or free memory (unlike C or C++ with malloc/free or new/delete).
- The CLR tracks object lifetimes and cleans up unused objects automatically.

2. Type Safety

- Prevents unsafe casts or memory access violations.
- Ensures objects are only used in valid ways, reducing bugs and security issues.

3. Security

- Runs in a **sandboxed environment** with **code access security** and **verification**.
- Protects against unauthorized operations, especially important for code from external or unknown sources.

4. Exception Handling

- Provides structured exception handling (try, catch, finally).
- The CLR manages the call stack and unwinds it safely during exceptions.

5. Just-In-Time Compilation (JIT)

- C# code is first compiled into **Intermediate Language (IL)**.
- At runtime, the CLR uses JIT to compile IL into native machine code.
- Enables platform independence at compile time and optimization at runtime.

6. Cross-Language Interoperability

- Managed code written in C#, VB.NET, or F# can interact because they all compile to IL and run on the CLR.
- Promotes code reuse and integration across different .NET languages.

Key Similarities Between struct and class

Both struct and class in C#:

- Can have **fields, properties, methods, constructors**, and **interfaces**.
- Support **encapsulation** and **access modifiers**.
- Are **user-defined types**.

This similarity in structure and syntax might make a struct seem "like a class" at first glance.

Key Differences Between struct and class in C#

Feature	**struct**	**class**
Type Category	Value type	Reference type
Memory Allocation	Typically on the stack	On the heap
Inheritance	Cannot inherit from other types	Can inherit from other classes
Default Constructor	Cannot define a parameterless one	Can define any constructor
Boxing	Boxing occurs when cast to object	No boxing needed
Performance	More efficient for small data	Better for complex, large objects
Nullability	Cannot be null unless nullable	Can be null

Why Struct Is Not Just a Class

Although struct may **look like** a class, it behaves very differently because:

- It's a **value type**: When passed or assigned, the **entire value is copied**.
- It has **no inheritance hierarchy** beyond System.ValueType.
- It's used for **small, lightweight objects** that do not require reference semantics.