CSharp 10.0 Programming

Start learning how to program using the C# programming language and become a professional software developer from a beginner/novice level.

About the Course & Importance of C#

C# is a popular and powerful object-oriented programming language developed by Microsoft. It is widely used to build desktop applications, web applications, Web APIs, cloud solutions, mobile apps, games, enterprise software, and microservices. Whether you are a beginner or an experienced developer, learning C# is a valuable step in your software development journey.

This course is designed for students, fresh graduates, working professionals, and developers who want to build a strong foundation in programming using the latest C# 14.0 features. It starts with the basics and gradually moves into advanced topics such as Object-Oriented Programming (OOP), Exception Handling, Collections, LINQ, Delegates, Events, Generics, Asynchronous Programming, File Handling, Reflection, and modern C# language features.

With its clean syntax, excellent tooling in Visual Studio, and seamless integration with the .NET ecosystem, C# makes software development simple and effective. By the end of this course, you will have the confidence and practical knowledge needed to write professional C# applications and prepare for technologies like ASP.NET Core, Web API, Microservices, Azure, MAUI, and Blazor.

Computer programming is really fun in general, and programming with C# Language using Visual Studio is even better & simpler!

Learning C# 14.0 :

There are 4 phases to learn C# 14.0.

1. Phase I – Introduction to .NET Core & Language Fundamentals
2. Phase II – Application Development and C# 14.0 new features
3. Phase III (Files , Database's, XML & Cloud Storage)
4. Phase IV (Distributed Programming)

Pre-Requisites: NA. Duration : 45 days
Course Materials & Textbooks : 1. Running Notes in class, 2. Pdfs, 3. HandOuts, 4. Certification Material in Pdfs will be provided for this course.
Reference Websites:www.dotnetgurukul.com
Instructor: Praveen Kumar M Learning from Praveen Sir, is always different and you will experience it within few sessions – you attend and work regularly as per given guidance then you will be best in any team that you work in any company – assured.

ASP.NETCore Course Syllabus by Praveen

1. Phase I – Introduction to .NET Core & Language Fundamentals -🎯 Available Free on YouTube

Topic	Key Concepts Covered
.NET Core & Languages	• Getting Started with .NET: From .NET Framework to .NET Core• Brief history of Microsoft & evolution of programming (COM/DCOM, C, C++, VB, VC++)• Overview of .NET Framework• CLR, FCL, Assemblies (EXE/DLL)• CLR Execution Model (C# → IL → CLR → Execution)• Features: Interoperability, Security, Scalability• Introduction to .NET Core (Cross-platform, Open Source, Modular)• .NET Core vs .NET Framework
Detailed Exploration of .NET Core Components	• CoreCLR – Runtime Execution & Memory Management• CoreFX – Base Class Libraries• RyuJIT – JIT Compilation & Performance Benefits• .NET Standard – API Compatibility• Unified .NET Ecosystem (.NET 5+)• .NET CLI (dotnet new, build, run, publish)• NuGet Package Management• Garbage Collector (GC) & Generational GC• Key Features of .NET Core
.NET Evolution and Versions	• .NET Framework Versions & Compatibility• Supported C# & Visual Studio Versions• Legacy vs Modern .NET Development• .NET Core Versions (1.0 → .NET 8)• Evolution to Unified .NET (.NET 5+)• C# Versions & Feature Highlights (LINQ, async/await, pattern matching, records)• Visual Studio Evolution & Compatibility
C# Data Types and Memory Management	• Built-in C# Data Types• Global, Stack & Heap Memory• Stack vs Heap Allocation• Common Type System (CTS)• Common Language Specification (CLS)
Value Types, Reference Types & Type Casting	• Value Types vs Reference Types• Memory Behavior & Copy Semantics• Stack & Heap Storage Structures• Implicit & Explicit Type Casting• Parse() & TryParse()• Nullable Casting & **as** Keyword• Checked & Unchecked Blocks
Working with Arrays – Basics	• Introduction to Arrays• Declaration & Initialization• Default Values & Collection Initializers• Accessing Array Elements• Iteration using for, while & foreach• Array Properties & Methods (Length, Sort, Reverse, IndexOf)
Working with Arrays – Advanced	• Single-Dimensional Arrays• Multidimensional Arrays• Jagged Arrays• Accessing & Iterating 2D and Jagged Arrays
Understanding Methods and Parameters in C#	• Introduction to Methods• Method Signature & Definition• Return Types & Return Values• Arguments vs Parameters• Caller vs Called Methods• Pass by Value vs Pass by Reference• **ref**, **out**, and **in** Keywords

2. Phase II – Application Development and C# 14.0 new features

🔒 Premium Phase

Topic	Key Concepts Covered
Introduction to Object-Oriented Programming (OOP)	Class & Object, Properties, Methods, Access Modifiers, Encapsulation, Inheritance, Polymorphism, Abstraction (Overview), Procedural vs OOP, Creating Classes & Objects
OOP Principles – Deep Dive	Encapsulation (BankAccount), Inheritance (Vehicle, Car, Bike), Polymorphism (Shape & Draw), Abstraction (API Example)
Interfaces	Interface Fundamentals, Structure & Rules, Implementation Steps, Reusability, Testability, Dependency Injection (DI), Polymorphism, Separation of Concerns (SoC)
Abstract Classes	Abstract Classes, Abstract vs Non-Abstract Members, Rules, Reusability, Encapsulation, Polymorphism, Banking Example
Constructor Basics	Constructors, new Keyword, Constructor Execution Order, Inheritance Flow, Constructor Overloading, Constructor Chaining (this), Access Modifiers
Constructor Types	Default, Parameterized, Copy, Static & Private Constructors, Static Constructor Behavior, Constructor Exceptions, Best Practices
Destructors & Resource Management	Destructors, Garbage Collection Lifecycle, GC.Collect(), WaitForPendingFinalizers(), IDisposable, Dispose Pattern, using Statement
C# Class Types	Abstract Classes, Sealed Classes, Partial Classes, Static Classes, Real-world Use Cases
C# Method Types	Instance Methods, Static Methods, Extension Methods, Async Methods, Anonymous Methods
Delegates & Events	Delegates, Multicast Delegates, Events vs Delegates, Event Declaration & Raising, UI and File Download Scenarios
Properties in C#	Properties vs Fields, Read-only, Write-only, Read-Write Properties, Auto-Implemented Properties, Encapsulation Best Practices
Fields in C#	Instance Fields, Static Fields, Readonly Fields, Constants (const), Backing Fields
SOLID Principles – Part 1	OOP Recap, Importance of Design Principles, SRP, OCP, LSP, ISP, DIP, Monolithic vs SOLID Architecture
SOLID Principles – Part 2 (DIP & DI)	Dependency Inversion Principle, Dependency Injection, Constructor/Method/Property Injection, Built-in DI Container, IoC Containers (Autofac, Ninject), Docker & Kubernetes Overview
Design Patterns	Introduction to GoF Patterns, Singleton, Factory, Repository, Unit of Work, Microservices Architecture, Ocelot API Gateway
Exception Handling	Exceptions vs Errors, try-catch-finally, throw Keyword, using Statement, Custom Exceptions, Database Exception Handling
Async Programming & Multithreading	Concurrency Concepts, Threads, ThreadPool, Parallel Programming, Task-based Async Programming, Choosing Async Models
Reflection	Reflection Basics, Metadata Inspection, Type Class, Runtime Type Information, Dynamic Object Creation, Reflection Use Cases

3. Phase III (Files , Database' s, XML & Cloud Storage)

🔒 Premium Phase

Topic	Key Concepts Covered
Introduction to Generics	Recap of OOP Building Blocks (Interfaces, Classes, Methods, Properties, Fields), Why Generics?, Type Safety, Code Reusability, Performance (Avoid Boxing/Unboxing), Generic Classes (MyList<T>), Generic Interfaces (IRepository<T>)
Introduction to Collections & Types	What is a Collection?, Collections vs Arrays, Types of Collections (Index-based, Key-based, Sequential, Specialized), Generic vs Non-Generic Collections, Collection Characteristics, Access Patterns, Duplicates, Thread Safety, Common Interfaces (IEnumerable, ICollection, IDictionary, IList)
Deep Dive: IEnumerable, ICollection & IQueryable	Collection Interface Recap, Exploring Interfaces using Visual Studio Object Explorer, IEnumerable<T> & ICollection<T>, yield Keyword, IQueryable<T> vs IEnumerable<T>, EF Core & LINQ Query Execution, Deferred Execution
Iteration Patterns in C#	Iteration Fundamentals, for vs foreach, IEnumerable Basics, IEnumerator Mechanics, GetEnumerator(), MoveNext(), Current, Stateless vs Stateful Iteration
IComparable vs IComparer	Importance of Sorting, Sorting Primitive Types & Custom Objects, IComparable<T>, IComparer<T>, Multiple Sorting Strategies
Key-Based Collections	Hashtable, Dictionary, SortedDictionary, SortedList, Internal Storage (Hash Tables, Red-Black Trees, Sorted Arrays), Key-based Access, Ordering, Duplicates, Exceptions, GetHashCode()
Sequential / Priority-Based Collections	Stack (LIFO), Push, Pop, Peek, Queue (FIFO), Enqueue, Dequeue, Peek, Real-world Use Cases (Undo, Browser History, Order Processing)
Specialized Collections	ObservableCollection<T>, CollectionChanged Event, UI Data Binding, HashSet<T>, Duplicate Prevention, Set Operations, HybridDictionary, String Collections

4. Phase IV (Distributed Programming)

Topic	Key Concepts Covered
Introduction to ADO.NET & Database Connectivity Basics	Importance of Data-Driven Applications, Why ADO.NET Still Matters, Evolution of Microsoft Data Access (DAO, RDO, ADO → ADO.NET), ADO.NET Architecture (Connected vs Disconnected), .NET Data Providers (SqlClient, OleDb, Odbc), Connection Lifecycle & Best Practices, ADO.NET vs EF Core
SQL Connection Pooling & Database Connection Best Practices	Connection Pooling Architecture, Benefits of Pooling, Connection String Settings (Pooling, Min/Max Pool Size, Timeout), Connection Lifecycle, Performance Optimization
Executing SQL Commands Using ADO.NET	SqlCommand Overview, ExecuteNonQuery(), ExecuteReader(), ExecuteScalar(), ExecuteXmlReader(), Async Methods (ExecuteReaderAsync(), ExecuteNonQueryAsync()), CRUD Operations

Connected Data Access with SqlDataReader	SqlDataReader Fundamentals, Forward-Only Read-Only Result Sets, Read(), HasRows, GetValue(), Access by Column Index & Name, Mapping Data to Objects, Async Data Reader
Disconnected Data Access – DataSet & DataTable	Connected vs Disconnected Architecture, DataSet, DataTable, DataAdapter, DataRow, RowState, In-Memory CRUD Operations, Persisting Changes, AcceptChanges(), Error Handling
Disconnected Data Access – Multiple Tables & Relationships	Working with Multiple Tables, DataRelation, Parent-Child Relationships, Foreign Key & Unique Constraints, Master-Detail Data Binding, DataView for Sorting & Filtering
Advanced DataSet Handling & Serialization	DataSet Serialization, XML, JSON & CSV Formats, WriteXml(), ReadXml(), Newtonsoft.Json, Data Exchange Best Practices
Parameterized Queries & Stored Procedures	SQL Injection Attacks, Parameterized Queries, Add() vs AddWithValue(), Stored Procedures, Database Security Best Practices
Understanding & Using MARS (Multiple Active Result Sets)	Limitations of SqlDataReader, Multiple Active Result Sets (MARS), Enabling MARS, Real-world Parent–Child Query Scenarios
Transactions in ADO.NET	Importance of Transactions, ACID Properties, SqlTransaction, BeginTransaction(), Commit(), Rollback(), Error Handling
Asynchronous Programming in ADO.NET	Synchronous vs Asynchronous Database Calls, async/await Recap, Async ADO.NET APIs, SqlBulkCopy.WriteToServerAsync(), UI Responsiveness
Bulk Data Loading with SqlBulkCopy	Introduction to SqlBulkCopy, Destination Table Requirements, Supported Data Sources, Column Mapping, High-Performance Bulk Inserts
Stored Procedure Input, OUTPUT & RETURN Values	Stored Procedure Fundamentals, Input Parameters, OUTPUT Parameters, RETURN Values, Executing Stored Procedures from ADO.NET