C Sharp Syllabus

C Sharp (C#)

Introduction to C#

- ✓ **Overview of C#:** History, features, and benefits of using C#.
- ✓ **Setting Up Environment:** Installing Visual Studio or .NET SDK, creating a C# project.
- ✓ **Writing the First Program:** Hello World program and understanding the program structure.

Basic Syntax and Fundamentals

- ✓ **Basic Syntax**: Understanding namespaces, using directive, and main method.
- ✓ **Data Types and Variables**:
    - Primitive types (int, float, double, char, bool).
    - Non-primitive types (string, object).
    - Nullable types.
- ✓ **Constants and Literals**: Defining constants using the const keyword.
- ✓ **Type Casting**: Implicit and explicit type conversion.

Operators and Control Flow

- ✓ **Operators:**
  - Arithmetic, relational, logical, bitwise, assignment, and conditional operators.
- ✓ **Decision-Making Statements**:
  - if, else if, else, switch-case.
- ✓ **Loops**:
  - for, while, do-while, and foreach loops.
  - **Control Statements**: break, continue, goto.

Arrays and Collections

- ✓ **Arrays**: Single-dimensional, multi-dimensional, and jagged arrays.
- ✓ **Collections**:
  - List, Dictionary, HashSet, Queue, and Stack.
  - Understanding generic collections (List<T>, Dictionary<K, V>).

Object-Oriented Programming (OOP)

- ✓ **Classes and Objects**: Defining classes, creating objects, accessing members.
- ✓ **Constructors**: Default, parameterized, and static constructors.
- ✓ **Inheritance**: Base and derived classes, method overriding, base keyword.
- ✓ **Encapsulation**: Access modifiers (public, private, protected, internal), getters and setters.
- ✓ **Polymorphism**: Method overloading, method overriding, and dynamic polymorphism.
- ✓ **Abstraction**: Abstract classes and interfaces, implementation in C#.
- ✓ **Static Members**: Static fields, methods, and constructors.
- ✓ **Partial Classes**: Splitting a class definition across multiple files.

Advanced C# Features

- ✓ **Delegates and Events:**
  - Understanding delegates, multicast delegates, and events.
- ✓ **Lambda Expressions**: Writing inline anonymous functions.
- ✓ **LINQ (Language Integrated Query):**
  - Basic LINQ syntax, querying collections, filtering, sorting, and grouping data.
- ✓ **Extension Methods**: Adding methods to existing types.
- ✓ **Indexers**: Defining and using indexers in custom classes.
- ✓ **Attributes**: Predefined attributes, custom attributes.

Error and Exception Handling

- ✓ **Try-Catch Blocks**: Handling runtime errors.
- ✓ **Finally Block**: Cleanup operations after exceptions.
- ✓ **Throwing Exceptions**: Custom exceptions and re-throwing exceptions.
- ✓ **Creating Custom Exceptions**: User-defined exception classes.

Working with Files and I/O

- ✓ **File Handling:** Reading and writing files using StreamReader and StreamWriter.
- ✓ **Directories**: Creating, deleting, and navigating directories.
- ✓ **Serialization and Deserialization**: Binary and XML serialization.

Multithreading and Asynchronous Programming

- ✓ **Threads**: Creating and managing threads, thread synchronization.
- ✓ **Tasks**: Using Task class for asynchronous operations.
- ✓ **Async and Await**: Writing asynchronous methods for improved application performance.
- ✓ **Parallel Programming**: Introduction to parallel processing.

Windows Application Development

- ✓ **WinForms**: Creating graphical user interfaces using Windows Forms.
- ✓ **WPF (Windows Presentation Foundation)**: Basics of WPF, data binding, and MVVM architecture.
- ✓ **Event Handling**: Understanding and implementing event-driven programming.

Web Development with C#

- ✓ **ASP.NET Core**: Introduction to building web applications.
- ✓ **MVC Architecture**: Model-View-Controller pattern for scalable applications.
- ✓ **Web APIs**: Creating RESTful services using ASP.NET Core.
- ✓ **Blazor Framework**: Building interactive web UIs using C#.

Database Interaction

- ✓ **ADO.NET**: Connecting to databases, executing queries, and handling transactions.
- ✓ **Entity Framework**: ORM framework for database interaction.
    - Code-First and Database-First approaches.
- ✓ **LINQ to SQL**: Querying databases using LINQ.

Memory Management

- ✓ **Garbage Collection**: Automatic memory management in C#.
- ✓ **Dispose and Finalize**: Cleaning up unmanaged resources.
- ✓ **IDisposable Interface**: Implementing custom cleanup logic.

Advanced Topics

- ✓ **Reflection**: Inspecting and invoking code metadata at runtime.
- ✓ **Dynamic Programming**: Using dynamic type for runtime binding.
- ✓ **Generics**: Understanding and implementing generic classes and methods.
- ✓ **Dependency Injection**: Basics of DI and its importance in software design.

Testing and Debugging

- ✓ **Unit Testing**: Writing tests using frameworks like MSTest, NUnit, or xUnit.
- ✓ **Debugging Tools**: Using Visual Studio’s debugging features.
- ✓ **Code Analysis**: Refactoring and analyzing code for best practices.