C# and .NET Core Course Outline

Beginner to Advanced

This course outline is designed to take you from a complete beginner to an advanced C# and .NET Core developer. The program is structured into three progressive levels: Beginner, Intermediate, and Advanced, with practical projects to solidify your skills at each stage. Each module includes key concepts, tools, and recommended projects to ensure hands-on learning.

Level 1: Beginner

Goal: Build a solid foundation in C# programming fundamentals and create basic console and desktop applications.

Module 1: Introduction to Programming and .NET Ecosystem

Concepts:

- What is programming and software development
- Introduction to .NET ecosystem (.NET Core, .NET Framework, .NET 5+)
- Understanding compilation and runtime
- IDE setup (Visual Studio, VS Code, Rider)
- Solution and project structure

Skills:

- Setting up development environment
- Creating your first console application
- Understanding the build process

Duration: 1 week
Project: "Hello World" console application with basic user interaction

Module 2: C# Fundamentals

Concepts:

- Variables, data types, and constants
- Operators (arithmetic, logical, comparison)
- Control structures (if/else, switch)
- Loops (for, while, foreach)
- Methods and parameters

- Arrays and basic collections
- String manipulation
- Exception handling basics (try/catch)

Skills:

- Writing clean, readable code
- Problem-solving with algorithms
- Basic debugging techniques

Duration: 3 weeks
Project: Calculator console application with menu system

Module 3: Object-Oriented Programming (OOP)

Concepts:

- Classes and objects
- Constructors and destructors
- Properties and fields
- Methods and method overloading
- Encapsulation and access modifiers
- Inheritance and polymorphism
- Abstract classes and interfaces
- Static members and static classes

Skills:

- Designing class hierarchies
- Implementing interfaces
- Understanding object relationships

Duration: 3 weeks
Project: Library Management System (console-based)

Module 4: Collections and Generics

Concepts:

- Generic collections (List<T>, Dictionary<TKey, TValue>)
- Arrays vs. collections

- LINQ basics (Where, Select, OrderBy)
- Iterators and yield
- Generic methods and classes
- Constraints on generics

Skills:

- Choosing appropriate data structures
- Basic data manipulation with LINQ
- Working with generic types

Duration: 2 weeks
Project: Student Grade Management System

Module 5: File I/O and Data Persistence

Concepts:

- File and directory operations
- Reading and writing text files
- Working with streams
- Serialization (JSON, XML)
- Basic database concepts
- SQLite integration

Skills:

- Persisting application data
- File system navigation
- Data serialization/deserialization

Duration: 2 weeks
Project: Personal Expense Tracker with file storage

Module 6: Version Control and Development Practices

Concepts:

- Git fundamentals
- Repository management
- Branching and merging

- Code commenting and documentation
- Basic debugging and testing
- Code organization and naming conventions

Tools: Git, GitHub, Visual Studio debugger

Duration: 1 week
Project: Refactor previous projects with proper Git workflow

Level 2: Intermediate

Goal: Build dynamic web applications and APIs using ASP.NET Core with database integration.

Module 7: Advanced C# Features

Concepts:

- Delegates and events
- Lambda expressions and functional programming
- Advanced LINQ operations
- Async/await programming
- Nullable reference types
- Pattern matching
- Records and init-only properties
- Extension methods

Skills:

- Functional programming concepts
- Asynchronous programming patterns
- Modern C# syntax

Duration: 2 weeks
Project: Asynchronous file processor with event notifications

Module 8: Database Programming

Concepts:

- Relational database fundamentals
- SQL basics (SELECT, INSERT, UPDATE, DELETE)

- Entity Framework Core
- Code-First vs Database-First approaches
- Migrations and database versioning
- Repository pattern
- Unit of Work pattern

Tools: SQL Server, SQLite, Entity Framework Core

Duration: 3 weeks
Project: Blog application with EF Core

Module 9: ASP.NET Core Web APIs

Concepts:

- HTTP protocol and REST principles
- ASP.NET Core architecture
- Controllers and actions
- Model binding and validation
- Dependency injection
- Middleware pipeline
- Configuration management
- Logging and error handling
- API documentation with Swagger

Skills:

- Building RESTful APIs
- Request/response handling
- API versioning
- Authentication basics

Duration: 3 weeks
Project: Task Management API with CRUD operations

Module 10: Web Development with ASP.NET Core MVC

Concepts:

- MVC pattern and architecture

- Views and Razor syntax
- Model binding and validation
- Partial views and view components
- Layout pages and sections
- Client-side validation
- Working with forms
- Session and state management

Skills:

- Building web applications
- Server-side rendering
- Form handling and validation

Duration: 3 weeks
Project: E-commerce product catalog website

Module 11: Testing and Quality Assurance

Concepts:

- Unit testing fundamentals
- Test-driven development (TDD)
- Mocking and test doubles
- Integration testing
- Code coverage
- Testing best practices

Tools: xUnit, NUnit, Moq, FluentAssertions

Duration: 2 weeks
Project: Add comprehensive tests to previous projects

Level 3: Advanced

Goal: Master enterprise-level development with microservices, cloud deployment, and advanced architectural patterns.

Module 12: Advanced ASP.NET Core Features

Concepts:

- Custom middleware development
- Background services and hosted services
- SignalR for real-time communication
- gRPC services
- Health checks and monitoring
- Caching strategies (in-memory, distributed)
- Rate limiting and throttling

Duration: 2 weeks
Project: Real-time chat application with SignalR

Module 13: Authentication and Authorization

Concepts:

- Authentication vs. authorization
- JWT tokens and claims-based authentication
- OAuth 2.0 and OpenID Connect
- Identity framework
- Role-based and policy-based authorization
- API security best practices
- CORS and security headers

Duration: 2 weeks
Project: Secure API with user authentication and role-based access

Module 14: Microservices and Distributed Systems

Concepts:

- Microservices architecture patterns
- Service communication (HTTP, messaging)
- API gateways and service discovery
- Circuit breaker pattern
- Event-driven architecture
- Message queues (RabbitMQ, Azure Service Bus)
- Distributed caching (Redis)

Tools: Docker, RabbitMQ, Redis

Duration: 3 weeks
Project: Microservices-based e-commerce system

Module 15: Cloud Development and DevOps

Concepts:

- Cloud platforms (Azure, AWS)
- Containerization with Docker
- Container orchestration (Kubernetes basics)
- CI/CD pipelines
- Infrastructure as Code
- Monitoring and logging in production
- Performance optimization
- Scalability patterns

Tools: Docker, Azure/AWS, GitHub Actions, Application Insights

Duration: 3 weeks
Project: Deploy microservices to cloud with CI/CD pipeline

Module 16: Performance and Optimization

Concepts:

- Performance profiling and benchmarking
- Memory management and garbage collection
- Async best practices
- Database optimization
- Caching strategies
- Load testing
- Application monitoring

Tools: BenchmarkDotNet, dotMemory, Application Insights

Duration: 1 week
Project: Performance optimization of existing applications

Module 17: Advanced Topics and Emerging Technologies

Concepts:

- Blazor for web development
- .NET MAUI for cross-platform apps
- Machine Learning with ML.NET
- GraphQL APIs
- Event sourcing and CQRS
- Clean Architecture patterns
- Domain-Driven Design (DDD)

Duration: 2 weeks
Project: Choose one advanced technology and build a proof-of-concept

Module 18: Portfolio and Career Preparation

Concepts:

- Building a professional portfolio
- Open-source contributions
- Code review best practices
- System design principles
- Interview preparation
- Soft skills for developers

Skills:

- Showcasing projects on GitHub
- Writing technical documentation
- Preparing for technical interviews
- Understanding system architecture

Duration: 1 week
Project: Complete portfolio website with deployed applications

Course Summary

Total Duration: ~32 weeks (8 months at 25-30 hours/week)

Learning Path:

- **Beginner:** Master C# fundamentals, OOP, and basic applications (12 weeks)

- **Intermediate:** Learn web development, APIs, and databases (13 weeks)
- **Advanced:** Dive into microservices, cloud, and enterprise patterns (7 weeks)

Projects: 15+ projects, from simple console applications to complex distributed systems

Key Technologies Covered:

- C# and .NET Core/.NET 6+
- ASP.NET Core Web API and MVC
- Entity Framework Core
- SQL Server/SQLite
- Docker and containerization
- Cloud platforms (Azure/AWS)
- Testing frameworks
- CI/CD pipelines

Outcome: By the end, you'll have a professional portfolio, deployable applications, and skills to land entry-level to senior-level .NET developer roles in enterprise environments.

Recommended Learning Resources

Books:

- "C# in Depth" by Jon Skeet
- "Clean Code" by Robert C. Martin
- "Microservices Patterns" by Chris Richardson

Online Platforms:

- Microsoft Learn
- Pluralsight
- Udemy
- YouTube (Nick Chapsas, Tim Corey)