

## C# 3.0 Programming in the .NET Framework (MS50150)

**Duration:** 5 days

**Course Description:** This course provides students with the knowledge and skills to develop applications in the .NET Framework 3.5 using the C# 3.0 programming language. The C# 3.0 revision introduces new productivity, performance, functional programming, and convenience features into the language. This course features an overview of all language-related features, as well as an introduction to general .NET Framework features such as garbage collection, assembly loading, Reflection, Language-Integrated Query (LINQ) and many others

**Audience:** This course is intended for developers who wish to extend their skills into C# 3.0 and .NET.

**Prerequisites:** Working knowledge of an object-oriented programming language (C++ preferred) and familiarity with object-oriented design principles.

### MODULE 1: INTRODUCTION TO THE .NET FRAMEWORK

This module explains how to develop applications in a variety of languages for the .NET Framework, and how various runtime mechanisms facilitate the execution of managed programs.

- Introduction to the .NET Framework
- Common Language Runtime Components – Garbage collector (GC), Common Type System (CTS), Just-in-Time compiler (JIT)
- An Overview of Managed Languages
- Microsoft Intermediate Language (IL)
- Native Image Generator (NGEN)
- An Overview of the Framework Class Library (FCL)
- .NET Version Evolution – from .NET 1.0 to .NET 3.5

### MODULE 2: INTRODUCTION TO C# 3.0

This module explains how to make the first steps in the Visual Studio Integrated Development Environment (IDE) and use the Framework Class Library (FCL) to develop simple C# applications.

- C# 3.0: Overview and Design Goals
- The Visual Studio Integrated Development Environment
- "Hello World" in C#

- Namespaces and References – Importing types, multi-targeting support, target platform
- Console Operations
- String Formatting
- Disassembling .NET – ILDASM, .NET Reflector

#### Lab: Basic Operations

- Simple console operations
- String output formatting

### MODULE 3: THE .NET TYPE SYSTEM

This module explains how to choose and use the proper category of types – reference types or value types – for the task at hand, how to convert between different types and be wary of performance penalties introduced by boxing and unboxing.

- The Common Type System
- The Common Language Specification
- Primitives and Built-in Types
- Value Types and Reference Types
- Boxing and Unboxing
- System.Object Class Members
- Type Conversions

#### Lab: Reviewing Reference Types and Value Types

- Comparing operations on value types and reference types

## C# 3.0 Programming in the .NET Framework (MS50150)

### Lab: Reviewing Object Equality

- Comparing equality operations on value types and reference types

### MODULE 4: C# CLASSES

This module explains how to design and implement C# classes with a variety of member types, and accommodate for the coding guidelines of .NET Framework types.

- Class Members
- Access Modifiers
- Nested Types
- Fields
- Constructors and Static Constructors
- Constants and Readonly Fields
- Properties and Automatic Properties
- Object Initializer Syntax
- Methods and Static Methods
- Static Classes
- Extension Methods
- Partial Types and Partial Methods
- The new Operator
- Parameter Modifiers
- Variable Parameter Lists
- The Entry Point and its Parameters
- Destructors

### Lab: Basic Class

- Rectangle class – methods, static methods, fields, properties
- Linked list, partial methods, and extension methods

### MODULE 5: GARBAGE COLLECTION

This module explains how to interact with the .NET garbage collector (a service that automatically reclaims unused memory), and how to use finalization to execute cleanup code for unmanaged resources.

- Destructor and Finalization
- Tracing Garbage Collection
- Interacting with the Garbage Collector
- Generations
- Weak References

### MODULE 6: XML DOCUMENTATION

This module explains how to document code while developing it and how to generate professional-looking external documentation from XML comments.

- XML Overview
- XML Documentation in Comments
- Auxiliary Tools – Sandcastle, DocumentX!

### MODULE 7: ARRAYS AND STRINGS

This module explains how to declare and use arrays and strings.

- Array Definition and Usage – Multi-dimensional, jagged, System.Array
- Casting and Enumerating Arrays
- String Class Members
- String Immutability
- StringBuilder
- String Literals

### Lab: Name Processing

- Reading, sorting, and writing strings and files

### MODULE 8: OBJECT ORIENTED PROGRAMMING IN C#

This module explains how to use inheritance and polymorphism in C# classes, including up- and down-casts. Lessons

- Inheritance and Polymorphism
- Up Casts and Down Casts
- Inheritance and Overriding Subtleties

### Lab: Shapes

- Shape inheritance hierarchy
- Extending the hierarchy – a compound shape (Composite design pattern)

### MODULE 9: STRUCTURES AND ENUMERATIONS

This module explains how to implement user-defined value types (structures) in .NET applications with the motivation for doing so, and how to design enumeration types for convenient usage.

## C# 3.0 Programming in the .NET Framework (MS50150)

- User-Defined Value Types
- Field Initialization
- Nullable Types
- Enumerations and Flags

### MODULE 10: INDEXERS

This module explains how to implement indexed class properties emulating array access syntax.

- Indexers
- Consuming Indexers from Other .NET Languages

#### Lab: Receptionist Scheduling

- Indexer access to classes
- Multi-parameter indexers

### MODULE 11: EXCEPTION HANDLING

This module explains how to design error-reporting using exceptions in managed applications, how to throw, catch, and handle exceptions in a resource-oriented environment, and how to declare user-defined exceptions.

- Error Reporting Alternatives
- Throwing and Catching Exceptions
- Exception Types and Objects
- Inner Exceptions
- User-Defined Exceptions
- Resource Management
- Checked and Unchecked Arithmetic
- Exception Design Guidelines and Performance

#### Lab: Incorporating Exception Handling

- Adding exception handling

### MODULE 12: INTERFACES

This module explains how to declare interfaces, how to implement them explicitly or implicitly, and how to use system interfaces that are part of the .NET Framework.

- Interface Declaration and Implementation
- Explicit Interface Implementation
- System Interfaces

- Extending Interfaces using Extension Methods

#### Lab: Enumeration Capabilities

- Providing enumeration via foreach
- Providing find (with a comparer) capabilities to the class from Lab 4

### MODULE 13: OPERATOR OVERLOADING

This module explains how to add user-defined operators to types, in order to provide a more convenient syntactic usage form.

- Overloading Operators
- Operator Names in the CLS
- User-Defined Conversions – Implicit and explicit, sequence of conversions

### MODULE 14: DELEGATES AND EVENTS

This module explains how to declare and define delegates as multi-function pointers, how delegates are implemented, how to use anonymous methods (closures) for improving programming productivity, and how to use events to implement common design patterns.

- Delegate Definition and Usage
- Delegate Implementation
- Multi-cast Delegates
- Anonymous Methods
- Lambda Functions
- Events
- Event Design Patterns

#### Lab: Sorting with Delegates

- Sort criteria implementation using delegates

#### Lab: Event-Based Chat System

- Client and server event-based chat

### MODULE 15: PREPROCESSOR DIRECTIVES

This module explains how to use preprocessor directives to conditionally compile code into C# applications.

- Preprocessing Directives

## C# 3.0 Programming in the .NET Framework (MS50150)

- Defining and Undefined Preprocessor Directives

### MODULE 16: IMPROVED C++

This module explains how to avoid common pitfalls when transitioning from C++ to C#.

- Control Flow Statements
- Switch Blocks

### MODULE 17: METADATA AND REFLECTION

This module explains how to use Reflection to obtain run-time information about types, methods, properties and fields, and how to create object instances and interact with them at run-time without requiring early-binding during compilation.

- Metadata Tables
- Reflection Types
- System.Activator

#### Lab: Self-Registration with Interfaces

- Self-registered singleton repository using a marker interface

### MODULE 18: ATTRIBUTES

This module explains how to decorate code elements with framework-defined and custom attributes, how to design and implement custom attribute types, how to query attributes using Reflection and how to design aspect-oriented applications using attributes.

- Attribute Class
- Attribute Examples
- Applying Attributes
- User-Defined Attributes and Attribute Usage
- Querying Attributes with Reflection

#### Lab: Logging with Attributes

- Primitive object serialization for logging purposes

#### Lab: Self-Registration with Attributes

- Self-registration (see Lab 12) with attributes instead of a marker interface

### MODULE 19: GENERICS

This module explains how to design and implement generic types and methods for the widest range of data types, how to use constraints to limit the application of generic code, how to interrogate generic types at run-time using Reflection and how .NET generics compare to C++ templates.

- Motivation for Generics
- Generic Constraints
- Generic Interfaces, Methods, and Delegates
- .NET Generics vs. C++ Templates
- Generics and Reflection

### MODULE 20: GENERIC COLLECTIONS

This module explains how to use the system generic collections to obtain better performance with value types and reference types, and how to use generic system interfaces.

- Built-in Generic Collections
- Generic System Interfaces
- Collection Initializers

#### Lab: Implementing a Generic Collection

- Implementing IList on a collection

### MODULE 21: DEPLOYMENT, VERSIONING, AND CONFIGURATION

This module explains how to deploy, version, configure and register .NET assemblies in a private or shared configuration scenario, how to control versioning and binding policy through application configuration files, and how to create multi-module (and even multi-language) assemblies.

- Deployment and Versioning of .NET Assemblies
- Private and Shared Assemblies – The Global Assembly Cache (GAC)
- Application Configuration Files
- Versioning Policies
- Friend Assemblies

## C# 3.0 Programming in the .NET Framework (MS50150)

- Multi-Module Assemblies

### Lab: Creating and Registering Assemblies

- Creating a privately deployed assembly
- Using probing configuration to access an assembly at a sub-directory
- Registering a shared assembly in the GAC
- Controlling versioning (binding) policy using application configuration

### Lab: Using LINQ

- Implementing extension methods
- Implementing custom query operators
- Implementing the query pattern
- Writing declarative LINQ queries against object models

### MODULE 22: UNSAFE CODE AND INTEROPERABILITY

This module explains how to use the .NET interoperability features to integrate managed and unmanaged code within the same application, and how to use unsafe code (C# pointers) to obtain performance and interoperability benefits.

- .NET Interoperability Options
- Introduction to Platform Invoke (P/Invoke)
- Unsafe Code – C# Pointers

### Lab: Calling Exported C Functions from C#

- Calling a custom exported C function from C
- Calling a Win32 API (requiring a reverse P/Invoke callback)

### MODULE 23: INTRODUCTION TO LANGUAGE-INTEGRATED QUERY (LINQ)

This module explains how to use Language-Integrated Query (LINQ) constructs and associated language features to increase productivity and model declarative data-driven applications instead of implementing them in the standard imperative manner.

- Anonymous Types and Implicit Variables
- Expression Trees
- Query Operators and the Query Pattern
- Language-Integrated Query Keywords and Query Translation
- LINQ to Objects