# 10266 - Programming in C# with Microsoft Visual Studio 2010

Duration: **5 days**

## Overview:

The course focuses on C# program structure, language syntax, and implementation detailswith .NET Framework 4.0. This course describes the new enhancements in the C# 4.0 language by using Visual Studio 2010. In this course, lower-intermediate level programmers gain the knowledge and skills they need to develop C# applications for the Microsoft .NET Framework 4.0. The course highlights the structure of C# 4.0 programs, language syntax, and implementation details. This course is not mapped to any exam.

## Target Audience:

This course is intended for experienced developers who already have programming experience in C, C++, Visual Basic, or Java and understand the concepts of object-oriented programming.This course is not designed for new programmers; it is targeted at professional developers with at least 12 months experience of programming in an object-oriented environment.

## Pre-requisites:

Before attending this course, students must have:

- At least 12 months experience working with an Object Oriented language
- Have C++ or Java knowledge:
- Creating Classes
- Inheritance and Abstraction
- Polymorphism
- Interfaces
- Exceptions
- Knowledge of the Visual Studio IDE.

## At Course Completion:

After completing this course, students will be able to:

- Explain the purpose of the .NET Framework, and understand how to use C# and Visual Studio 2010 to build .NET Framework applications.
- Understand the syntax of

## Module 1: Introducing C# and the .NET Framework

**Lessons**
- Introduction to the .NET Framework
- Creating Projects Within Visual Studio 2010
- Writing a C# Application
- Building a Graphical Application
- Documenting an Application
- Running and Debugging Applications by Using Visual Studio 2010

**Lab : Introducing C# and the .NET Framework**
- Building a Simple Console Application
- Building a WPF Application
- Verifying the Application
- Generating Documentation for an Application

## Module 2: Using C# Programming Constructs

**Lessons**
- Declaring Variables and Assigning Values
- Using Expressions and Operators
- Creating and Using Arrays
- Using Decision Statements
- Using Iteration Statements

**Lab : Using C# Programming Constructs**
- Calculating Square Roots with Improved Accuracy
- Converting Integer Numeric Data to Binary
- Multiplying Matrices

## Module 3: Declaring and Calling Methods

**Lessons**
- Defining and Invoking Methods
- Specifying Optional Parameters and Output Parameters

**Lab : Declaring and Calling Methods**
- Calculating the Greatest Common Divisor of Two Integers by Using Euclid's Algorithm
- Calculating the GCD of Three, Four, or Five Integers
- Comparing the Efficiency of Two Algorithms
- Displaying Results Graphically
- Solving Simultaneous Equations (optional)

## Module 4: Handling Exceptions

**Lessons**
- Handling Exceptions
- Raising Exceptions

**Lab : Handling Exceptions**
- Making a Method Fail-Safe
- Detecting an Exceptional Condition
- Checking for Numeric Overflow

## Module 5: Reading and Writing Files

**Lessons**
- Accessing the File System
- Reading and Writing Files by Using Streams

**Lab : Reading and Writing Files**
- Building a Simple Editor
- Making the Editor XML Aware

## Module 6: Creating New Types

**Lessons**
- Creating and Using Enumerations
- Creating and Using Classes
- Creating and Using Structs
- Comparing References to Values

**Lab : Creating New Types**
- Using Enumerations to Specify Domains
- Using a Struct to Model a Simple Type
- Using a Class to Model a More Complex Type
- Using a Nullable Struct

## Module 7: Encapsulating Data and Methods

**Lessons**
- Controlling Visibility of Type Members
- Sharing Methods and Data

**Lab : Encapsulating Data and Methods**
- Hiding Data Members
- Using Static Members to Share Data
- Implementing an Extension Method

## Module 8: Inheriting From Classes and Implementing Interfaces

**Lessons**
- Using Inheritance to Define New Reference Types
- Defining and Implementing Interfaces
- Defining Abstract Classes

**Lab : Inheriting From Classes and Implementing Interfaces**
- Defining an Interface
- Implementing an Interface
- Creating an Abstract Class

## Module 9: Managing the Lifetime of Objects and Controlling Resources

**Lessons**
- Introduction to Garbage Collection
- Managing Resources

**Lab : Managing the Lifetime of Objects and Controlling Resources**
- Implementing the IDisposable Interface
- Managing Resources Used By an Object

basic C# programming constructs.
- Create and call methods in a C# application.
- Catch, handle and throw exceptions.
- Perform basic file IO operations in a C# application.
- Create and use new types (enumerations, classes, and structures), and understand the differences between reference types and value types.
- Control the visibility and lifetime of members in a type.
- Use inheritance to create new reference types.
- Manage the lifetime of objects and control the use of resources.
- Define properties and indexers to encapsulate data, and define operators for this data.
- Decouple an operation from the method that implements an operation, and use these decoupled operations to handle asynchronous events.
- Use collections to aggregate data, and use Generics to implement type-safe collection classes, structures, interfaces, and methods.
- Implement custom collection classes that support enumeration.
- Query in-memory data by using LINQ.
- Integrate code written by using a dynamic language such as Ruby and Python, or technologies such as COM, into a C# application

CODE:0-0-MSM10266-ILT

## Module 10: Encapsulating Data and Defining Overloaded Operators

**Lessons**
- Creating and Using Properties
- Creating and Using Indexers
- Overloading Operators

**Lab : Creating and Using Properties**
- Defining Properties in an Interface
- Implementing Properties in a Class
- Using Properties Exposed By a Class

**Lab : Creating and Using Indexers**
- Implementing an Indexer to Access Bits in a Control Register
- Using an Indexer Exposed by a Class

**Lab : Overloading Operators**
- Defining the Matrix and MatrixNotCompatible Types
- Implementing Operators for the Matrix Type
- Testing the Operators for the Matrix Type

## Module 11: Decoupling Methods and Handling Events

**Lessons**
- Declaring and Using Delegates
- Using Lambda Expressions
- Handling Events

**Lab : Decoupling Methods and Handling Events**
- Raising and Handling Events
- Using Lambda Expressions to Specify Code

## Module 12: Using Collections and Building Generic Types

**Lessons**
- Using Collections
- Creating and Using Generic Types
- Defining Generic Interfaces and Understanding Variance
- Using Generic Methods and Delegates

**Lab : Using Collections**
- Optimizing a Method by Caching Data

**Lab : Building Generic Types**
- Defining a Generic Interface
- Implementing a Generic Interface
- Implementing a Test Harness for the BinaryTree Project
- Implementing a Generic Method

## Module 13: Building and Enumerating Custom Collection Classes

**Lessons**
- Implementing a Custom Collection Class
- Adding an Enumerator to a Custom Collection Class

**Lab : Building and Enumerating Custom Collection Classes**
- Implementing the IList TItem Interface
- Implementing an Enumerator by Writing Code
- Implementing an Enumerator by Using an Iterator

## Module 14: Using LINQ to Query Data

**Lessons**
- Using the LINQ Extension Methods and Query Operators
- Building Dynamic LINQ Queries and Expressions

**Lab : Using LINQ to Query Data**
- Using the LINQ Query Operators
- Building Dynamic LINQ Queries

## Module 15: Integrating Visual C# Code with Dynamic Languages and COM Components

**Lessons**
- Integrating C# Code with Ruby and Python
- Accessing COM Components from C#

**Lab : Integrating C# Code with Dynamic Languages and COM Components**
- Integrating Code Written by Using a Dynamic Language into a C# Application
- Using a COM Component from Visual C# Application