Programming in C# - MOC On Demand (MS-20483)

Modality: On Demand

Duration: 2 Days

SATV Value: 2

This course sets you up for the Exam pf 70-483 prompting the Certification of 70-483. This course does exclude the Official Exam Voucher, in any case, you can demand to buys the Official Exam Voucher independently.

About this course:

This course is MS Official popular course open for 90 days from the date of course demand if you have a yearly membership, or course buys separately. Course access will lapse following 90 days enlistment of the course.

This instructional class shows engineers the programming abilities that are required for designers to make Windows applications utilizing the language of C#. During this course, understudies survey the nuts and bolts of the language syntax, structure of C# program, language syntax, and implementation details, and afterward unite their insight during the time as they make an application that includes many features of the .NET Framework 4.5.

The course presents a considerable lot of technologies and techniques utilized by enterprise applications and modern desktop, including:

- Using remote data.
- Programming the user interface.
- Handling events.
- Building new data types.
- Integrating with unmanaged code.
- Accessing a database.
- Creating custom attributes.
- Performing operations asynchronously.
- Encrypting and decrypting data.

Toward the finish of the course, understudies should leave the class with strong information on C# and how to utilize it to create .NET Framework 4.5 applications.

This course utilizes Visual Studio 2012, functioning on Windows 8.

The normal compensation for a C# designer is **$88,856** every year.

Course Objective:

- Use legacy to make a class chain of command, expand a class of .NET Framework, and

make generic classes and strategies.
- Explain the main features and syntax of C#.
- Utilize the kinds in the WCF Data Services and System.Net namespace to access and question remote information.
- Execute the fundamental structure and basic components of a normal desktop application.
- Make classes, characterize and execute interfaces, and make and utilize generic collections.
- Improve the reaction time and throughput of apps by utilizing tasks and asynchronous operations.
- Write and read information by utilizing record output/input and streams, and deserialize and serialize information in various formats.
- Encode and decode information by utilizing asymmetric and symmetric encryption.
- Make and utilize a substance information model for getting to a database and use LINQ to inquiry and update information.
- Look at the metadata of types by utilizing reflection, make and utilize custom characteristics, create code at runtime, and oversee get assembly versions.
- Develop a graphical UI by utilizing XAML.
- Catch and handle exceptions, create and call methods, and explain the observing prerequisites of enormous scope applications.
- Incorporate dynamic components and unmanaged libraries into a C# application.

Audience:

This course is intended for:

Experienced engineers who already have experience in programming with C, JavaScript, C++, Microsoft Visual Basic®, Objective-C, or Java and comprehend the ideas of object-oriented programming.

Prerequisites:

Designers going to this course should as of now have increased some constrained experience utilizing C# to finish essential programming assignments. All the more explicitly, understudies ought to have hands-on experience utilizing C# that shows their comprehension of the following:

- How to declare, name, assign and initialize values to variables inside an application.
- How to use arithmetic operators to do the calculations of arithmetic involving including at least one factors;
- The way to make the syntax of the code for simple statements of programming utilizing C# language keywords and understand errors of syntax using the Visual Studio IDE.
- How to use logical operators to consolidate expressions that contain relational operators.
- How to use relational operators to realize the connection between two factors or expressions;
- How to sort data in a loop.
- The way to utilize the Visual Studio IDE to find out simple logic errors.
- How to make an easy structure for branching utilizing an IF statement.
- How to make a task that accepts arguments (returns and parameters the value of a specified type).

Suggested prerequisites courses:

Introduction to the Databases of SQL - MOC on-Demand (MS-10985)

Programming for Absolute Beginners

Course Outline:

Module 1: Review of C# Syntax

This module reviews the core syntax and features of the C# programming language. It also provides an introduction to the Visual Studio 2012 debugger.

Lessons

- Overview of Writing Applications using C#
- Datatypes, Operators, and Expressions
- C# Programming Language Constructs

Lab : Developing the Class Enrolment Application

After completing this module, students will be able to:

- Describe the architecture of .NET Framework applications and use the features that Visual Studio 2012 and C# provide to support .NET Framework development.
- Use the basic data types, operators, and expressions provided by C#.
- Use standard C# programming constructs.

Module 2: Creating Methods, Handling Exceptions, and Monitoring Applications

This module explains how to create and call methods, catch and handle exceptions. This module also describes the monitoring requirements of large-scale applications.

Lessons

- Creating and Invoking Methods
- Creating Overloaded Methods and Using Optional and Output Parameters
- Handling Exceptions
- Monitoring Applications

Lab : Extending the Class Enrolment Application Functionality

After completing this module, students will be able to:

- Create and invoke methods, pass parameters to methods, and return values from methods.
- Create overloaded methods, and use optional parameters and output parameters.
- Catch and handle exceptions and write information to the event log.
- Explain the requirement for implementing logging, tracing, and profiling when building large-scale applications.

Module 3: Developing the Code for a Graphical Application

This module describes how to implement the basic structure and essential elements of a typical desktop application, including using structures and enumerations, collections, and events.

Lessons

- Implementing Structs and Enums
- Organizing Data into Collections
- Handling Events

Lab : Writing the Code for the Grades Prototype Application

After completing this module, students will be able to:

- Define and use structures and enumerations.
- Create and use simple collections for storing data in-memory.
- Create, subscribe to, and raise events.

Module 4: Creating Classes and Implementing Type-safe Collections

This module explains how to create classes, define and implement interfaces, and create and use generic collections. This module also describes the differences between value types and reference types in C#.

Lessons

- Creating Classes
- Defining and Implementing Interfaces
- Implementing Type-safe Collections

Lab : Adding Data Validation and Type-safety to the Grades Application

After completing this module, students will be able to:

- Create and use custom classes.
- Define and implement custom interfaces.
- Use generics to implement type-safe collections.

Module 5: Creating a Class Hierarchy by Using Inheritance

This module explains how to use inheritance to create a class hierarchy and extend a .NET Framework class. This module also describes how to create generic classes and define extension methods.

Lessons

- Creating Class Hierarchies

- Extending .NET Framework Classes
- Creating Generic Types

Lab : Refactoring Common Functionality into the User Class

After completing this module, students will be able to:

- Define abstract classes and inherit from base classes to create a class hierarchy.
- Inherit from .NET Framework classes and use extension methods to add custom functionality to the inherited class.
- Create generic classes and methods.

Module 6: Reading and Writing Local Data

This module explains how to read and write data by using file input/output (I/O) and streams, and how to serialize and deserialize data in different formats.

Lessons

- Reading and Writing Files
- Serializing and Deserializing Data
- Performing I/O Using Streams

Lab : Generating the Grades Report

After completing this module, students will be able to:

- Read and write data to and from the file system by using file I/O.
- Convert data into a format that can be written to or read from a file or other data source.
- Use streams to send and receive data to or from a file or other data source.

Module 7: Accessing a Database

This module explains how to create and use an entity data model for accessing a database, and how to use LINQ to query and update data.

Lessons

- Creating and Using Entity Data Models
- Querying Data by Using LINQ
- Updating Data by Using LINQ

Lab : Retrieving and Modifying Grade Data

After completing this module, students will be able to:

- Create an entity data model, describe the key classes contained in the model, and customize the generated code.

- Use LINQ to query and work with data.
- Use LINQ to insert, update, and delete data.

Module 8: Accessing Remote Data

This module explains how to use the types in the System.Net namespace, and WCF Data Services, to query and modify remote data.

Lessons

- Accessing Data Across the Web
- Accessing Data in the Cloud

Lab : Retrieving and Modifying Grade Data in the Cloud

After completing this module, students will be able to:

- Use the classes in the System.Net namespace to send and receive data across the Web.
- Create and use a WCF Data Service to access data in the cloud.

Module 9: Designing the User Interface for a Graphical Application

This module explains how to build and style a graphical user interface by using XAML. This module also describes how to display data in a user interface by using data binding.

Lessons

- Using XAML to Design a User Interface
- Binding Controls to Data
- Styling a User Interface

Lab : Customizing Student Photographs and Styling the Application

After completing this module, students will be able to:

- Define XAML views and controls to design a simple graphical user interface.
- Use XAML data binding techniques to bind XAML elements to a data source and display data.
- Add styling and dynamic transformations to a XAML user interface.

Module 10: Improving Application Performance and Responsiveness

This module explains how to improve the throughput and response time of applications by using tasks and asynchronous operations.

Lessons

- Implementing Multitasking by using Tasks and Lambda Expressions
- Performing Operations Asynchronously

- Synchronizing Concurrent Access to Data

Lab : Improving the Responsiveness and Performance of the Application

After completing this module, students will be able to:

- Create tasks and lambda expressions to implement multitasking.
- Define and use asynchronous methods to improve application responsiveness.
- Coordinate concurrent access to data shared across multiple tasks by using synchronous primitives and concurrent collections.

Module 11: Integrating with Unmanaged Code

This module explains how to integrate unmanaged libraries and dynamic components into a C# application. This module also describes how to control the lifetime of unmanaged resources.

Lessons

- Creating and Using Dynamic Objects
- Managing the Lifetime of Objects and Controlling Unmanaged Resources

Lab : Upgrading the Grades Report

After completing this module, students will be able to:

- Integrate unmanaged code into a C# application by using the Dynamic Language Runtime.
- Control the lifetime of unmanaged resources and ensure that they are disposed properly.

Module 12: Creating Reusable Types and Assemblies

This module explains how to examine the metadata of types by using reflection, create and use custom attributes, generate managed code at runtime, and manage different versions of assemblies.

Lessons

- Examining Object Metadata
- Creating and Using Custom Attributes
- Generating Managed Code
- Versioning, Signing and Deploying Assemblies

Lab : Specifying the Data to Include in the Grades Report

After completing this module, students will be able to:

- Examine the metadata of objects at runtime by using reflection.
- Create and use custom attribute class.
- Generate managed code at runtime by using CodeDOM.
- Manage different versions of an assembly and deploy an assembly to the Global Assembly

Cache.

Module 13: Encrypting and Decrypting Data

This module explains how to encrypt and decrypt data by using symmetric and asymmetric encryption.

Lessons

- Implementing Symmetric Encryption
- Implementing Asymmetric Encryption

Lab : Encrypting and Decrypting Grades Reports

After completing this module, students will be able to:

- Perform symmetric encryption by using the classes in the System.Security namespace.
- Perform asymmetric encryption by using the classes in the System.Security namespace.