

MENTORI
LEARNING CENTER

Programming in C#

Course #	Exam:	Prerequisites	Technology:	Delivery Method:	Length:
20483	70-483		Visual Studio	Instructor-led (classroom)	**5 Days**

Overview

About this Course

This training course teaches developers the programming skills that are required for developers to create Windows applications using the C# language. During their five days in the classroom students review the basics of C# program structure, language syntax, and implementation details, and then consolidate their knowledge throughout the week as they build an application that incorporates several features of the .NET Framework 4.5.

The course introduces many of the techniques and technologies employed by modern desktop and enterprise applications, including:

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

At the end of the course, students should leave the class with a solid knowledge of C# and how to use it to develop .NET Framework 4.5 applications.

This course uses Visual Studio 2012, running on Windows 8.

Audience Profile

This course is intended for experienced developers who already have programming experience in C, C++, JavaScript, Objective-C, Microsoft Visual Basic, or Java and understand the concepts of object-oriented programming.

This course is not designed for students who are new to programming; it is targeted at professional developers with at least one month of experience programming in an object-oriented environment.

At Course Completion

After completing this course, students will be able to:

- Describe the core syntax and features of C#.
- Create and call methods, catch and handle exceptions, and describe the monitoring requirements of large-scale applications.
- Implement the basic structure and essential elements of a typical desktop application.
- Create classes, define and implement interfaces, and create and use generic collections.
- Use inheritance to create a class hierarchy, extend a .NET Framework class, and create generic classes and methods.
- Read and write data by using file input/output and streams, and serialize and deserialize data in different formats.

- Create and use an entity data model for accessing a database and use LINQ to query and update data.
- Use the types in the System.Net namespace and WCF Data Services to access and query remote data.
- Build a graphical user interface by using XAML.
- Improve the throughput and response time of applications by using tasks and asynchronous operations.
- Integrate unmanaged libraries and dynamic components into a C# application.
- Examine the metadata of types by using reflection, create and use custom attributes, generate code at runtime, and manage assembly versions.
- Encrypt and decrypt data by using symmetric and asymmetric encryption.

Prerequisites

Developers attending this course should already have gained some limited experience using C# to complete basic programming tasks. More specifically, students should have hands-on experience using C# that demonstrates their understanding of the following

- How to name, declare, initialize and assign values to variables within an application.
- How to use:
- arithmetic operators to perform arithmetic calculations involving one or more variables;
- relational operators to test the relationship between two variables or expressions;
- Logical operators to combine expressions that contain relational operators.
- How to create the code syntax for simple programming statements using C# language keywords and recognize syntax errors using the Visual Studio IDE.
- How to create a simple branching structure using an IF statement.
- How to create a simple looping structure using a For statement to iterate through a data array.
- How to use the Visual Studio IDE to locate simple logic errors.
- How to create a Function that accepts arguments (parameters and returns a value of a specified type.
- How to design and build a simple user interface using standard controls from the Visual Studio toolbox.
- How to connect to a SQL Server database and the basics of how to retrieve and store data.
- How to sort data in a loop.
- How to recognize the classes and methods used in a program.

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

- Implementing Edit Functionality for the Students List
- Implementing Insert Functionality for the Students List
- Implementing Delete Functionality for the Students List
- Displaying the Student Age

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

- Refactoring the Enrolment Code
- Validating Student Information
- Saving Changes to the Class List

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

- Adding Navigation Logic to the Application
- Creating Data Types to Store User and Grade Information
- Displaying User and Grade Information

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

- Implementing the Teacher, Student, and Grade Types as Classes
- Adding Data Validation to the Grade Class
- Displaying Students in Name Order
- Enabling Teachers to Modify Class and Grade Data

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

- Creating and Inheriting from the User Base Class
- Implementing Password Complexity by Using an Abstract Method
- Creating the ClassFullException 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

- Serializing the Data for the Grades Report as XML
- Previewing the Grades Report
- Persisting the Serialized Grades Data to a File

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

- Creating an Entity Model from the The School of Fine Arts Database
- Updating Student and Grade Data Using the Entity Framework
- Extending the Entity Model to Validate 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

- Creating a WCF Data Service for the SchoolGrades Database
- Integrating the WCF Data Service into the Application
- Retrieving Student Photographs Over the Web (if time permits)

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

- Customizing the Appearance of Student Photographs
- Styling the Logon View
- Animating the StudentPhoto Control (If Time Permits)

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

- Ensuring that the User Interface Remains Responsive When Retrieving Data for Teachers
- Providing Visual Feedback During Long-Running Operations

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

- Generating the Grades Report by Using Microsoft Office Word
- Controlling the Lifetime of Word Objects by Implementing the Dispose Pattern

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

- Creating the IncludeInReport Attribute
- Generating the Report
- Storing the Grades.Utilities Assembly Centrally

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

- Encrypting the Grades Report
- Decrypting the Grades Report

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