

# Programming with C#

Course 2124—Five days—Instructor-led

## Introduction

The goal of this course is to provide students with the knowledge and skills they need to develop C# applications for the Microsoft .NET Platform. The course focuses on C# program structure, language syntax, and implementation details.

C# was created to be the programming language best suited for writing enterprise applications for .NET. C# combines the high productivity of Microsoft Visual Basic with the raw power of C++. It is a simple, object-oriented, and type-safe programming language that is based on the C and C++ family of languages.

## Audience

This course is intended for experienced developers who already have programming experience in C, C++, Visual Basic, or Java. These developers will be likely to develop enterprise business solutions.

## At Course Completion

At the end of the course, students will be able to:

- List the major elements of the .NET Framework and explain how C# fits into the .NET Platform.
- Analyze the basic structure of a C# application and be able to document, debug, compile, and run a simple application.
- Create, name, and assign values to variables.
- Use common statements to implement flow control, looping, and exception handling.
- Create methods (functions and subroutines) that can return values and take parameters.
- Create, initialize, and use arrays.
- Explain the basic concepts and terminology of object-oriented programming.
- Use common objects and reference types.
- Create, initialize, and destroy objects in a C# application.
- Build new C# classes from existing classes.
- Create self-contained classes and frameworks in a C# application.
- Define operators, use delegates, and add event specifications.
- Implement properties and indexers.
- Use predefined and custom attributes.

## Prerequisites

Before attending this course, students must have:

- Experience with programming in C, C++, Visual Basic, Java, or another programming language.

- Familiarity with the Microsoft .NET strategy as described on the Microsoft .NET Web site: <http://www.microsoft.com/net/>.

Familiarity with the .NET Framework as described on the MSDN Magazine Web site:

- <http://msdn.microsoft.com/msdnmag/issues/0900/Framework/Framework.asp>
- <http://msdn.microsoft.com/msdnmag/issues/1000/Framework2/Framework2.asp>

## **Microsoft Certified Professional Exams**

There is no MCP exam associated with this course.

## **Student Materials**

The student kit includes a comprehensive workbook and other necessary materials for this class.

## **Course Outline**

### **Module 1: Overview of the Microsoft .NET Platform**

The following topics are covered in this module:

- Introduction to the .NET Platform
- Overview of the .NET Framework
- Benefits of the .NET Framework
- The .NET Framework Components
- Languages in the .NET Framework

After completing this module, you will be able to list the major elements of the .NET Framework and explain how the C# language fits into the .NET Platform. This includes:

- Describing the .NET Platform.
- Listing the main elements of the .NET Platform.
- Explaining the language support in the .NET Framework.
- Describing the .NET Framework and its components.

### **Module 2: Overview of C#**

The following topics are covered in this module:

- Structure of a C# Program
- Basic Input/Output Operations
- Recommended Practices
- Compiling, Running, and Debugging

After completing this module, you will be able to analyze the basic structure of a C# application and be able to document, debug, compile, and run a simple application. This includes:

- Explaining the structure of a simple C# program.
- Using the Console class of the System namespace to perform basic input/output operations.
- Handling exceptions in a C# program.
- Generating Extensible Markup Language (XML) documentation for a C# application.
- Compiling and executing a C# program.
- Using the Microsoft Visual Studio Debugger to trace program execution.

### **Module 3: Using Value-Type Variables**

The following topics are covered in this module:

- Common Type System
- Naming Variables
- Using Built-In Data Types
- Creating User-Defined Data Types
- Converting Data Types

After completing this module, you will be able to create, name, and assign values to variables. This includes:

- Describing the types of variables that you can use in C# applications.
- Naming your variables according to standard C# naming conventions.
- Declaring variables by using built-in data types.
- Assigning values to variables.
- Converting existing variables from one data type to another.
- Creating and using your own data types

### **Module 4: Statements and Exceptions**

The following topics are covered in this module:

- Introduction to Statements
- Using Selection Statements
- Using Iteration Statements
- Using Jump Statements
- Handling Basic Exceptions
- Raising Exceptions

After completing this module, you will be able to use common statements to implement flow control, looping, and exception handling. This includes:

- Describing the different types of control statements.
- Using jump statements.
- Using selection statements.
- Using iteration statements.
- Handling and raising exceptions.

## **Module 5: Methods and Parameters**

The following topics are covered in this module:

- Using Methods
- Using Parameters
- Using Overloaded Methods

After completing this module, you will be able to create methods (functions and subroutines) that can return values and take parameters. This includes:

- Creating static methods that accept parameters and return values.
- Passing parameters to methods in different ways.
- Declaring and using overloaded methods.

## **Module 6: Arrays**

The following topics are covered in this module:

- Overview of Arrays
- Creating Arrays
- Using Arrays

After completing this module, you will be able to create, initialize, and use arrays. This includes:

- Creating, initializing, and using arrays of varying rank.
- Using command-line arguments in a C# program.
- Describing the relationship between an array variable and an array instance.
- Using arrays as parameters for methods.
- Returning arrays from methods.

## **Module 7: Essentials of Object-Oriented Programming**

The following topics are covered in this module:

- Classes and Objects

- Using Encapsulation
- C# and Object Orientation
- Defining Object-Oriented Systems

After completing this module, you will be able to explain the basic concepts and terminology of object-oriented programming. This includes:

- Defining the terms object and class in the context of object-oriented programming.
- Describing the three core aspects of an object: identity, state, and behaviour.
- Describing abstraction and how it helps you to create reusable classes that are easy to maintain.
- Using encapsulation to combine methods and data in a single class and enforce abstraction.
- Explaining the concepts of inheritance and polymorphism.
- Creating and using classes in C#.

### **Module 8: Using Reference-Type Variables**

The following topics are covered in this module:

- Using Reference-Type Variables
- Using Common Reference Types
- The Object Hierarchy
- Namespaces in the .NET Framework
- Data Conversions

After completing this module, you will be able to use common objects and reference types. This includes:

- Describing the key differences between reference types and value types.
- Using common reference types such as string.
- Explaining how the object type works and becoming familiar with the methods it supplies.
- Describing common namespaces in the .NET Framework.
- Determining whether different types and objects are compatible.
- Explicitly and implicitly converting data types between reference types.
- Performing boxing and unboxing conversions between reference and value data.

### **Module 9: Creating and Destroying Objects**

The following topics are covered in this module:

- Using Constructors
- Initializing Data
- Objects and Memory
- Resource Managements

After completing this module, you will be able to create, initialize, and destroy objects in a C# application. This includes:

- Using constructors to initialize objects.
- Creating overloaded constructors that can accept varying parameters.
- Describing the lifetime of an object and what happens when it is destroyed.
- Creating destructors and using Finalize methods.

## **Module 10: Inheritance in C#**

The following topics are covered in this module:

- Deriving Classes
- Implementing Methods
- Using Sealed Classes
- Using Interfaces
- Using Abstract Classes

After completing this module, you will be able to build new C# classes from existing classes. This includes:

- Deriving a new class from a base class and calling members and constructors of the base class from the derived class.
- Declaring methods as virtual and override or hiding them as required.
- Sealing a class so that it cannot be derived from.
- Implementing interfaces by using both the implicit and explicit methods.
- Describing the use of abstract classes and their implementation of interfaces

## **Module 11: Aggregation, Namespaces, and Advanced Scope**

The following topics are covered in this module:

- Using Internal Classes, Methods, and Data
- Using Aggregation
- Using Namespaces
- Using Modules and Assemblies

After completing this module, you will be able to create self-contained classes and frameworks in a C# application. This includes:

- Using internal access to allow classes to have privileged access to each other.
- Using aggregation to implement powerful patterns such as Factories.
- Using namespaces to organize classes.
- Creating simple modules and assemblies.

## **Module 12: Operators and Events**

The following topics are covered in this module:

- Introduction to Operators
- Operator Overloading
- Creating and Using Delegates
- Defining and Using Events

After completing this module, you will be able to define operators, use delegates, and add event specifications. This includes:

- Defining operators to make a class or struct easier to use.
- Using delegates to decouple a method call from a method implementation.
- Adding event specifications to a class to allow subscribing classes to be notified of changes in object state.

## **Module 13: Properties and Indexers**

The following topics are covered in this module:

- Using Properties
- Using Indexers

After completing this module, you will be able to implement properties and indexers. This includes:

- Creating properties to encapsulate data within a class.
- Defining indexers to gain access to classes by using array-like notation.

## **Module 14: Attributes**

The following topics are covered in this module:

- Overview of Attributes
- Defining Custom Attributes
- Retrieving Attribute Values

After completing this module, you will be able to use predefined and custom attributes. This includes:

- Using common predefined attributes.
- Creating simple custom attributes.
- Querying attribute information at run

time.