Microsoft Partner

CL_55339

Programming in C#

www.ked.com.mx

Por favor no imprimas este documento si no es necesario.

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 6.0. The course aims to follow the spirit of the Microsoft Official Curriculum course 20483, while bringing it completely up-to-date with the latest features of Visual Studio 2022, and the cross-platform capabilities of .NET 6.0.

Length.

5 Days.

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. Those new to programming should consider course CL_55337: Introduction to Programming. The course uses C# as the language to facilitate an introduction to programming generally, whereas this course focuses on the C# language itself, making it an excellent follow-on course. If you want to learn to take full advantage of the C# language, then this is the course for you.

Prerequisites.

Developers attending this course should already have gained some limited experience using C# to complete basic

- The CL_55337 course uses C# as the language to facilitate an introduction to programming.
- Course CL_55339 focuses on the C# language itself, making it an excellent follow-on course.

At course completion.

After completing this course, students will be able to:

- Explain how to use Visual Studio 2022 to create and run a Web application.
- Describe the new features of HTML5, and create and style HTML5 pages.
- Add interactivity to an HTML5 page by using JavaScript.
- Create HTML5 forms by using different input types, and validate user input by using HTML5 attributes and JavaScript code.
- Send and receive data to and from a remote data source by using XMLHTTP Request objects and Fetch API.
- Style HTML5 pages by using CSS.
- Create well-structured and easily-maintainable JavaScript code.
- Write modern JavaScript code and use babel to make it compatible to all browsers.
- Use common HTML5 APIs in interactive Web applications.
- Create Web applications that support offline operations.
- Create HTML5 Web pages that can adapt to different devices and form factors.
- Add advanced graphics to an HTML5 page by using Canvas elements, and by using and Scalable Vector Graphics.
- Enhance the user experience by adding animations to an HTML5 page.
- Use Web Sockets to send and receive data between a Web application and a server.
- Improve the responsiveness of a Web application that performs long-running operations by using Web Worker processes.
- Use WebPack to package web applications for production.

Exam.

None.

Course outline.

Module 1: C# Syntax.

Microsoft .NET 6 provides a comprehensive development platform that you can use to build, deploy, and manage applications and services. By using .NET, you can create visually compelling applications, enable seamless communication across technology boundaries, and provide support for a wide range of business processes.

In this module, you'll learn about some of the core features provided by.NET and Microsoft Visual Studio. You'll also learn about some of the core C# constructs that enable you to start developing .NET applications.

- Writing Applications in C# and .NET.
- Types of Data and Expressions.
- C# Language Constructs.

Lab: C# Syntax.

After completing this module, students will be able to:

- Write Applications in C# and .NET.
- Types of Data and Expressions.
- C# Language Constructs.

Module 2: C# Language Concepts.

Applications often consist of logical units of functionality that perform specific functions, such as providing access to data or triggering some logical processing. C# is an object-orientated language and uses the concept of methods to encapsulate logical units of functionality. Although a good practice is to have methods that do just one thing, they can be as simple or as complex as you like. It is also important to consider what happens to the state of your application when an exception occurs in a method.

- Methods.
- Method Overloading.
- Exception Handling.
- Monitoring.

Labs: C# Language Concepts.

- Methods.
- Method Overloading.
- Exception Handling.
- Monitoring.

After completing this module, students will be able to:

In this module, you'll learn how to create and use methods and how to handle exceptions. You'll also learn how to use logging and tracing to record the details of any exceptions that occur.

Module 3: C# Structures, Collections and Events.

To create effective applications you must first learn some fundamental C# constructs. You need to know how to create simple structures to represent the data items you are working with. You need to know how to organize these structures into collections, so that you can add items, retrieve items, and iterate over your items. Finally, you need to know how to subscribe to events so that you can respond to the actions of your users.

- Structs.
- Enums.
- Built-in Collections.
- Events.

Labs: C# Structures, Collections and Events.

- Structs.
- Enums.
- Built-in Collections.
- Events.

After completing this module, students will be able to:

- Create and use structs and enums.
- Organize data into collections.
- Create and subscribe to events.

Module 4: C# Classes.

In this module, you'll learn how to use interfaces and classes to define and create your own custom, reusable types. You'll also learn how to create and use enumerable type-safe collections of any type.

- Creating Classes.
- Interfaces.
- Understanding Generics in C#.

Labs: C# Classes.

- Creating Classes.
- Interfaces.
- Understanding Generics in C#.

After completing this module, students will be able to:

- Use interfaces and classes to define and create your own custom, reusable types.
- Create and use enumerable type-safe collections of any type.

Module 5: C# Inheritance.

In this module, you'll learn how to use inheritance to create class hierarchies and to extend .NET types.

- Hierarchies of Classes.
- Polymorphism.
- Extending Classes.

Labs: C# Inheritance.

- Hierarchies of Classes.
- Polymorphism.
- Extending Classes.

After completing this module, students will be able to:

- Use inheritance to create class hierarchies to extend .NET types.

Module 6: Input and Output.

In this module, you'll learn how to read and write data by using transactional filesystem I/O operations, how to serialize and deserialize data to the filesystem, and how to read and write data to the filesystem by using streams.

- File I/O.
- Serialization and Deserialization.
- Streams.

Labs: Input and Output.

- File I/O.
- Serialization and Deserialization.
- Streams.

After completing this module, students will be able to:

- Read and write data by using transaction filesystem I/O operations.
- How to searlize and deserialize data to the file system.
- How to read and write data to the filesystem by using streams.

Module 7: Database Access.

In this module, you'll learn how to use Entity Framework and how to query many types of data by using Language-Integrated Query (LINQ).

- Entity Framework.
- LINQ.

Labs: Database Access.

- Entity Framework.
- LINQ.

After completing this module, students will be able to:

- Use entity Framework.
- Learn how to query many types of data by using Language-Integrated Query (LINQ).

Module 8: Using the Network.

In this module, you'll learn how to use the request and response classes in the System.Net namespace to directly manipulate remote data sources. You'll also learn about REST and OData and look briefly at ASP.NET Core MVC.

- Web Services.
- REST and OData.
- ASP.NET Core MVC.

Labs: Using the Network.

- Web Services.
- REST and OData.
- ASP.NET Core MVC.

After completing this module, students will be able to:

- Use the request and response classes in the System.Net Namespace to directly manipulate remote data sources.

Module 9: Graphical User Interfaces.

In this module, you'll learn how to use Extensible Application Markup Language (XAML) and Windows Presentation Foundation (WPF) to create engaging UIs.

- Using UI Frameworks.
- Data binding.
- Styling the UI.

Labs: Graphical User Interfaces.

- Using UI Frameworks.
- Data binding.
- Styling the UI.

After completing this module, students will be able to:

- Use Extensible Application Markup Language (XAML) and Windows Presentation Foundation (WPF) to create engaging UIs.

Module 10: Application Performance.

In this module, you'll learn how to improve the performance of your applications by distributing your operations across multiple threads.

- Multitasking.
- Asynchronous Calls.
- Dealing with Conflicts.

Labs: Installing and Configuring Windows 7.

- Multitasking.
- Asynchronous Calls.
- Dealing with Conflicts.

After completing this module, students will be able to:

- Improve the performance of your applications by distributing your operations across multiple threads.

Module 11: C# Interop.

In this module, you'll learn how to interoperate with unmanaged code in your applications and how to ensure that your code releases any unmanaged resources.

- Dynamic Objects.
- Managing Resources.

Labs: C# Interop.

- Dynamic Objects.
- Managing Resources.

After completing this module, students will be able to:

- Interoperate with unmanaged code in your applications.
- Ensure that your code releases any unmanaged resources.

Module 12: Designing for Reuse.

In this module, you'll learn how to consume existing assemblies by using reflection, and how to add additional metadata to types and type members by using attributes. You'll also learn how to generate code at runtime by using the Code Document Object Model (CodeDOM) and how manage your .NET assemblies.

- Metadata.
- Attributes,
- Generating Code.
- Assemblies.

Labs: Designing for Reuse.

- Metadata.
- Attributes.
- Generating Code.
- Assemblies.

After completing this module, students will be able to:

- Consume existing assemblies by using reflection.
- Add additional metadata to types and type members using attributes.
- Generate code at runtime by using the Code Document Object Model (CodeDOM).
- Manage your .NET assemblies.