Course 10266A:

Programming in C#

Course Details

Course Outline

Module 1: Introducing C# and the .NET Framework

This module explains the .NET Framework, and using C# and Visual Studio 2010 for building .NET Framework applications.

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

After completing this module, students will be able to:

- Explain the purpose of the .NET Framework.

- Create Microsoft Visual C# projects by using Visual Studio 2010.

- Explain the structure of a C# application.

- Use the WPF Application template to build a simple graphical application.

- Use XML comments to document an application.

- Use the debugger to step through a program.

Module 2: Using C# Programming Constructs

This module explains the syntax of basic 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

After completing this module, students will be able to:

- Declare variables and assign values.

- Create expressions by using operators.

- Create and use arrays.

- Use decision statements.

- Use iteration statements.

Module 3: Declaring and Calling Methods

This module explains how to create and call 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)

After completing this module, students will be able to:

- Describe how to declare and call methods

- Define and call methods that take optional parameters and output parameters

Module 4: Handling Exceptions

This module explains how to catch exceptions and handle them. Students will also learn how to throw exceptions.

Lessons

- Handling Exceptions

- Raising Exceptions

Lab : Handling Exceptions

- Making a Method Fail-Safe

- Detecting an Exceptional Condition

- Checking for Numeric Overflow

After completing this module, students will be able to:
- Describe how to catch and handle exceptions

- Describe how to create and raise exceptions

Module 5: Reading and Writing Files

This module explains how to perform basic file I/O operations in a C# application.

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

After completing this module, students will be able to:
- Describe how to access the file system by using the classes that the .NET Framework provides.

- Describe how to read and write files by using streams.

Module 6: Creating New Types

This module explains how to create and use new types (enumerations, classes, and structures)

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

After completing this module, students will be able to:
- Describe how to create and use enumerations.

- Describe how to create and use classes.

- Describe how to create and use structures.

- Explain the differences between reference and value types.

Module 7: Encapsulating Data and Methods

This module explains how to control the visibility and lifetime of members in a type.

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

After completing this module, students will be able to:
- Describe how to control the visibility of type members.

- Describe how to share methods and data.

Module 8: Inheriting From Classes and Implementing Interfaces

This module explains how to use inheritance to create new reference types

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

After completing this module, students will be able to:
- Use inheritance to define new reference types.

- Define and implement interfaces.

- Define abstract classes.

Module 9: Managing the Lifetime of Objects and Controlling Resources

This module explains how to manage the lifetime of objects and control the use of 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

After completing this module, students will be able to:
- Describe how garbage collection works in the .NET Framework.

- Manage resources effectively in an application.

Module 10: Encapsulating Data and Defining Overloaded Operators

This module explains how to create properties and indexers to encapsulate data, and how to define operators for this data.

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

After completing this module, students will be able to:
- Explain how properties work and use them to encapsulate data.

- Describe how to use indexers to access data through an array-like syntax.

- Describe how to use operator overloading to define operators for your own types.

Module 11: Decoupling Methods and Handling Events

This module explains how to decouple an operation from the method that implements an operation, and how to use these decoupled methods to handle asynchronous 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

After completing this module, students will be able to:
- Describe the purpose of delegates, and explain how to use a delegate to decouple an operation from the implementing method.

- Explain the purpose of lambda expressions, and describe how to use a lambda expression to define an anonymous method.

- Explain the purpose of events, and describe how to use events to report that something significant has happened in a type that other parts of the application need to be aware of.

Module 12: Using Collections and Building Generic Types

This module introduces collections, and describes how to use Generics to implement type-safe collection classes, structures, interfaces, and methods.

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

After completing this module, students will be able to:

- Use collection classes.

- Define and use generic types.

- Define generic interfaces and explain the concepts of covariance and contravariance.

- Define and use generic methods and delegates.

Module 13: Building and Enumerating Custom Collection Classes

This module explains how to implement custom collection classes that support enumeration.

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

After completing this module, students will be able to:

- Implement a custom collection class.

- Define an enumerator for a custom collection class.

Module 14: Using LINQ to Query Data

This module explains how to query in-memory data by using LINQ.

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

After completing this module, students will be able to:

- Describe how to use the LINQ extension methods and query operators.

- Describe how to build dynamic LINQ queries and expressions.

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

This module explains how to integrate code written by using a dynamic language such as Ruby and Python, and technologies such as COM, into a C# application

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

After completing this module, students will be able to:

- Integrate Ruby and Python code into a Visual C# application.

- Invoke COM components and services from a C# application.