Course Code: **CIS161**
Course Name: Programming in C#
Certification: Microsoft Programming in C# - 70-483
Duration: 3 months
Tuition: $1895

Course Overview

Microsoft Visual C# is a principal development language for the .NET framework. Learn to manage program flow, use multithreading and asynchronous processing, and implement events, callbacks, and exception handling using C#. One of the main types in object-oriented programming is the class. Delve into classes and explore encapsulation, implementing a class hierarchy, using reflection, and managing the object life cycle. In addition to classes, there are a number of C# types and structures. Learn how to create and use types, convert value and ref types, and manipulate strings in C#. Debugging and securing your applications is crucial prior to deployment. Discover input validation, symmetric and asymmetric encryption, managing an assembly, using compiler directives, and implementing diagnostics. Data access and management is a critical aspect of most enterprise applications. Learn how to consume data, manipulate data and objects using LINQ, perform serialization, and use collections in C# applications. Discover how to implement threaded programs, use a Task Parallel Library and explore the differences between asynchronous and parallel programming. Discover how to work with Automated Memory Management, manage the Object Life Cycle and String operations. A critical aspect of most enterprise applications is data access. Discover how you can implement data access in your C# applications using IO operations and learn how to consume the data.

Course Content

- **Lesson 1** – **Programming in C#: Managing Program Flow**
  This lesson covers the following topics:
  - use the switch statement
  - program decisions using the If/Else statement
  - use the for and foreach iteration statements
  - use operators and evaluation expressions
  - use built-in delegate types to make code easier to create and read
  - create and use delegates
  - use lambda expressions
  - use anonymous methods
  - subscribe to an event
  - create event handlers
  - handle exceptions by implementing multiple catch blocks
  - use the finally block
  - create custom exception classes

- implement context-specific exception handling for SQL exceptions, communication exceptions, and others
- illustrate how to handle aggregate exceptions in multithreading

- **Lesson 2** – **Programming in C#: Creating Types**
  This lesson covers the following topics:
  - create, modify, and compile structs using fields, properties, and methods
  - create, modify, and debug enums
  - create and use classes
  - use constructors to instantiate classes
  - use optional and named parameters
  - create static class methods and variables
  - use extension methods
  - create and use indexers
  - specify the concept overloading and overriding
  - overload methods
  - override methods
  - illustrate the concept of generics
  - create and use generic type
  - use generic methods
  - convert value types
  - convert ref types
  - convert 'Value to Reference' and 'Reference to Value'
  - use the dynamic runtime
  - use IConvertible
  - use IFormattable

- **Lesson 3 – Programming in C#: Using Types**
  This lesson covers the following topics:
  - format strings
  - use auto-implemented properties
  - use public, internal, and private access modifiers
  - implement explicit interfaces
  - create interfaces
  - implement inheritance
  - create and implement indexers
  - use IEnumerable
  - use the Dispose method of the IDisposable interface to remove an object
  - define the concept of COM interoperability and IUnKnow interface
  - implement the IComparable interface

- use the System.Reflection namespace to provide a range of information about a type
- use types from the 'System.Reflection namespace' such as Assembly, PropertyInfo, and MethodInfo type
- demonstrate how to apply and read attributes
- create attribute classes to add to a code construct
- use CodeDom to create code dynamically

- **Lesson 4** – **Programming in C#: Debugging and Security Implementation**
  This lesson covers the following topics:
  - use regular expressions
  - validate data in the JSON format
  - use the SQLConnection and the SQLConnectionStringBuilder classes to guard against attacks on your database
  - use the AesCryptoServiceProvider class and the Common Language Runtime's CryptoStream object to implement encryption
  - use the Microsoft Windows Certificates Manager to locate Certification Authorities and manage user and local machine X.509 certificates
  - verify the integrity of data by using hashing on the contents of a file
  - use either faster symmetric or more secure asymmetric algorithms
  - generate a unique key pair and assign it to an assembly
  - create and use WinMD assembly
  - use Microsoft Visual Studio's Global Assembly Cache (GAC) to store and deploy strong-named components
  - use compiler directives
  - configure tracing functionality
  - work with trace switches and listeners
  - configure performance counters
  - write to the event log

- **Lesson 5** – **Programming in C#: Manipulating and Retrieving Data**
  This lesson covers the following topics:
  - access stored procedures from a model
  - create method-based and query expression-based LINQ queries
  - use the LINQ operator 'Where' for filtering
  - work with select and select many LINQ operations
  - query data using the operators projection, join, group, take, skip, and aggregate
  - use the LINQ to XML provider to load XML and loop through collections
  - use the LINQ to XML provider to parse attributes and query multiple elements
  - use XMLSerializer to serialize and deserialize XML
  - serialize and deserialize JSON data

- serialize and deserialize binary data
- distinguish typed and non-typed collections
- work with collections
- use the Dictionary object
- use the generic List object
- use the generic Queue object
- implement .NET interfaces

- **Lesson 6 – Programming in C#: Managing Multithreating**
  This lesson covers the following topics:
  - describe the differences between thrown and re-thrown exceptions
  - using locking to prevent data from being accessed by multiple threads at a time
  - use synchronization events
  - use cancellation tokens to cancel a long-running task
  - demonstrate how to implement thread-safe methods to handle race conditions
  - use the interlocked class for thread-safe numeric access
  - perform asynchronous operation using task
  - use parallel invoke methods
  - distinguish between parallel for and for statements
  - use parallel ForEach methods
  - demonstrate how to spawn threads using ThreadPool
  - use task to unblock the user interface thread
  - speed up LINQ queries using Parallel LINQ (PLINQ)
  - manage data using ConcurrentBag collection
  - demonstrate how to facilitate asynchronous pattern usage for Async and Await Keywords
  - schedule tasks using the task object
  - set up a series of tasks to run in a specified sequence

- **Lesson 7 – Programming in C#: Memory Management and String Operations**
  This lesson covers the following topics:
  - guide the behavior of the .NET Framework's garbage collector
  - manage the unmanaged resources using Idisposable
  - implement destructor
  - demonstrate how to implement Using Block
  - use the StringBuilder class to provide efficiencies when handling strings
  - manipulate strings using StringReader and StringWriter
  - use String methods
  - search for characters in a string and use the String.Remove method
  - use string interpolation
  - format strings

- using the CultureInfo object to format dates and numbers defined by the string.Format method

- **Lesson 8– Programming in C#: Implementing Data Access**
  This lesson covers the following topics:
  - use the FileStream class to read from and write to files on a file system
  - use the WebRequest class
  - parse network credentials safely
  - use GZIPStream
  - implement asynchronous programming with C#
  - use a LINQ query to select database data
  - use anonymous types in a select statement
  - update database records through the Entity Framework model
  - use LINQ query operators to operate on data sequences and query .NET arrays and collections
  - force LINQ query execution
  - consume data from a web service
  - create an extension method to add functionality to a class without extending it through inheritance