C# Programming Certification

Duration:
5 Days

What is the course about?
This training course teaches developers the programming skills that are required for 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. 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.

Duration
The course is 5 days full time.

Programming Experience
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.

Technical Skill
Developers attending this course should already have gained some limited experience using C# to complete basic programming tasks.

Private Training
This course can be offered privately to a group, team or company. A minimum of 4 delegates is required to schedule the course. The course can run onsite or on our premises. The prices for running the course on your site are R9 500 and R12 599 on our premises.

Public Training
The course is offered publicly. There is no set date for the course, the course is run on demand. The course needs a minimum of 4 delegates to run. We set tentative dates, the dates are confirmed once we have 4 confimred bookings for the course.

Course Topics

Implement Multithreading and Asynchronous Processing
Use the Task Parallel library (ParallelFor, Plinq, Tasks)
Create continuation tasks

Spawn threads by using ThreadPool
Unblock the UI
Use async and await keywords
Manage data by using concurrent collections

Manage Multithreading

Synchronize resources
Implement locking
Cancel a long-running task
Implement thread-safe methods to handle race conditions

Implement Program Flow

Iterate across collection and array items
Program decisions by using switch statements, if/then, and operators
Evaluate expressions

Creating and Implement Events and Callbacks

Create event handlers
Subscribe to and unsubscribe from events
Use built-in delegate types to create events
Create delegates
Lambda expressions
Anonymous methods

Implement Exception Handling

Handle exception types (SQL exceptions, network exceptions, communication exceptions, network timeout exceptions)
Catch typed vs. base exceptions
Implement try-catch-finally block
Throw exceptions
Determine when to rethrow vs. throw
Create custom exceptions

Create Types

Create value types (structs, enum), reference types, generic types, constructors, static variables, methods, classes, extension methods, optional and named parameters, and indexed properties
Create overloaded and overriden methods

Consume Types

Box or unbox to convert between value types
Cast types
Convert types
Handle dynamic types
Ensure interoperability with unmanaged code, for example, dynamic keyword

Enforce Encapsulation

Enforce encapsulation by using properties, by using accessors (public, private, protected), and by using explicit interface implementation

Create and implement a class hierarchy

Design and implement an interface

Inherit from a base class
Create and implement classes based on the IComparable, IEnumerable, IDisposable, and IUnknown
Interfaces

Find, Execute and Create Types at Runtime by Using Reflectiong

Create and apply attributes
Read attributes
Generate code at runtime by using CodeDom and lambda expressions
Use types from the System.Reflection namespace (Assembly, PropertyInfo, MethodInfo, Type)

Manage the object Lifecycle

Manage unmanaged resources
Implement IDisposable, including interaction with finalization
Manage IDisposable by using the Using statement
Manage finalization and garbage collection

Mainpulate Strings

Manipulate strings by using the StringBuilder, StringWriter, and StringReader classes
Search strings
Enumerate string methods
Format strings

Validate Application Input

Validate JSON data
Data collection types
Manage data integrity
Evaluate a regular expression to validate the input format
Use built-in functions to validate data type and content out of scope: writing regular expressions

Errors and Exceptions

What are Errors
Catching Errors and Exceptions
Raising Exceptions
Creating Your Own Exceptions
Throwing and Catching Exceptions

Perform Symmetric and Asymmetric Encryption

Choose an appropriate encryption algorithm
Manage and create certificates
Implement key management
Implement the System.Security namespace
Hashing data Encrypt streams

Manage Assemblies

Version assemblies
Sign assemblies using strong names
Implement side-by-side hosting

Put an assembly in the global assembly cache
Create a WinMD assembly

Debug an Application

Create and manage compiler directives
Choose an appropriate build type
Manage programming database files and symbols

Implement Diagnostics in an Application

Implement logging and tracing
Profiling applications
Create and monitor performance counters
Write to the event log

Perform I/O Operations

Read and write files and streams
Read and write from the network by using classes in the System.Net namespace
Implement asynchronous I/O operations

Consume Data

Retrieve data from a database
Update data in a database
Consume JSON and XML data
Retrieve data by using web services

Query and Manipulate Data and Objects by Using LINQ

Query data by using operators (projection, join, group, take, skip, aggregate)
Create method-based LINQ queries
Query data by using query comprehension syntax
Select data by using anonymous types
Force execution of a query
Read, filter, create, and modify data structures by using LINQ to XML

Serialize and Deserialize Data

Serialize and deserialize data by using binary serialization, custom serialization, XML Serializer, JSON Serializer, and Data Contract Serializer

Store Data in and Retrieve Data from Collections

Store and retrieve data by using dictionaries, arrays, lists, sets, and queues
Choose a collection type
Initialize a collection
Add and remove items from a collection
Use typed vs. non-typed collections
Implement custom collections
Implement collection interfaces