

70-483: Programming in C#

Candidates for this exam are developers with at least one year of experience programming essential business logic for a variety of application types, hardware, and software platforms using C#.

Candidates should also have a thorough understanding of the following:

- Managing program flow and events
- Asynchronous programming and threading
- Data validation and working with data collections including LINQ
- Handling errors and exceptions
- Working with arrays and collections
- Working with variables, operators, and expressions
- Working with classes and methods
- Decision and iteration statements

Objective Domain

Note: This document shows tracked changes that are effective as of December 14, 2017.

Manage Program Flow (25-30%)

Implement multithreading and asynchronous processing

Use the [Task Parallel library, including theParallel.For method, 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

Create 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, [including SQL exceptions, network exceptions, communication exceptions, network timeout exceptions](#); use catch statements; use base class of an exception; implement try-catch-finally blocks; throw exceptions; [rethrow an exception](#); create custom exceptions; [handle inner exceptions](#); [handle aggregate exceptions](#)

Create and Use Types (25-30%)

Create types

Create value types, [including structs and enum; create reference types, generic types, constructors, static variables, methods, classes, extension methods; create optional and named parameters;](#) create indexed properties; create overloaded and overridden methods

Consume types

Box or unbox to convert between value types; cast types; convert types; handle dynamic types; ensure interoperability with [unmanaged code that accesses COM APIs](#)

Enforce encapsulation

Enforce encapsulation by using properties; enforce encapsulation by using accessors, including public, private, protected, and [internal](#); enforce encapsulation 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 reflection

Create and apply attributes; read attributes; generate code at runtime by using CodeDom and [lambda Lambda](#) expressions; use types from the System.Reflection namespace, [including Assembly, PropertyInfo, MethodInfo, Type](#)

Manage the object life cycle

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

Manipulate strings

Manipulate strings by using the StringBuilder, StringWriter, and StringReader classes; search strings; enumerate string methods; format strings; [use string interpolation](#)

Debug Applications and Implement Security (25-30%)

Validate application input

Validate JSON data; [choose the appropriate data collection type;](#) manage data integrity; evaluate a regular expression to validate the input format; use built-in functions to validate data type and content

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 [preprocessor compiler](#) directives; choose an appropriate build type; manage [programming program](#) database files [and \(debug symbols\)](#)

Implement diagnostics in an application

Implement logging and tracing; profiling applications; create and monitor performance counters; write to the event log

Implement Data Access (25-30%)

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, including 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