Exam: C# Coding Specialist
Exam Link: https://knowledge-pillars.com/csharp-coding-specialist-certification/

Course: Intro to C# Applications

TEKS Code & Standard	TEKS Description	Corresponding CSC Specialist Objective(s)	Alignment Description
§130.280(c)(2)(A)	Demonstrate knowledge of operating systems, environments, and platforms.	1.1: Recognize and apply C# syntax; 1.2: Utilize C# data types	Understanding C# syntax, data types, and structure supports student familiarity with various programming environments and system requirements.
§130.280(c)(3)(A)	Demonstrate knowledge of programming design using modular components.	3.1: Working with Methods; 3.3: Method Overloading	C# Apprentice objectives emphasize defining and overloading methods, reinforcing modular programming design.
§130.280(c)(3)(B)	Design and develop programs using structured programming techniques.	2.1–2.5: Control Statements (if, switch, loops, jumps); 1.7: Use operators and precedence	Control statements and operator usage are key to structured programming, directly aligning with C# logic and flow objectives.
§130.280(c)(3)(C)	Write programs that use subprograms or methods.	3.1: Working with Methods; 3.2: Method Parameters	Focus on method declaration, parameter handling, and return types supports the development of reusable subprograms.
§130.280(c)(3)(D)	Use programming style conventions.	1.1: C# syntax and code structure; use of comments and IntelliSense	Syntax, code organization, and use of comments enforce professional and readable coding standards.
§130.280(c)(4)(A)	Demonstrate knowledge of object-oriented programming.	4.1–4.7: OOP Concepts including Classes, Inheritance, Encapsulation, Abstraction, and Polymorphism	Core OOP concepts such as encapsulation, polymorphism, and class structures are foundational topics in the C# Apprentice exam.
§130.280(c)(4)(B)	Create classes.	4.2: Class and Object; 4.3: Constructors	Students learn to define and use classes and constructors, aligning with TEKS class creation outcomes.
§130.280(c)(4)(C)	Create objects.	4.2: Class and Object	The C# Apprentice exam covers object instantiation and references as part of its object-oriented programming section.
§130.280(c)(4)(D)	Demonstrate the use of object-oriented principles, including encapsulation, inheritance, and polymorphism.	4.4–4.7: Inheritance, Encapsulation, Abstraction, Polymorphism	Advanced OOP concepts are explicitly covered in the exam and required by this TEKS standard.
§130.280(c)(5)(A)	Use appropriate operators, data types, and control structures.	1.2, 1.7: Data types and Operators; 2.1–2.5: Control Statements	Students gain fluency in operators, data structures, and flow control through related C# exam objectives.
§130.280(c)(5)(B)	Demonstrate input/output operations.	3.1: Methods (with return/parameters); 1.3: Variables; 1.2: Strings	I/O concepts are practiced using strings, variables, and method return values as per the exam.
§130.280(c)(5)(C)	Use exception handling techniques.	2.5: Use throw statements to raise exceptions	Handling errors with throw statements and structured exception logic matches exam content.
§130.280(c)(6)(A)	Use appropriate built-in functions and libraries.	3.1–3.3: Methods; 6.2: Lambda Expressions	Built-in method usage and LINQ foundations demonstrate the use of existing language features effectively.
§130.280(c)(6)(B)	Implement searching and sorting algorithms.	6.2: Lambda in LINQ (filter, sort); indirectly covered under LINQ introduction	LINQ and lambda expressions offer modern approaches to sorting and filtering, partially aligning to this requirement.
§130.280(c)(7)(A)	Apply debugging techniques.	1.1: IntelliSense; 2.5: throw/return statements	Use of IntelliSense and exception handling supports development of debugging strategies.
§130.280(c)(7)(B)	Test program performance.	6.2: LINQ/Lambda optimizations (efficiency is implied in usage of collections)	LINQ queries and method efficiency form a foundation for performance considerations.
§130.280(c)(8)(A)	Apply industry-recognized software development and project management methodologies.	5.1–5.4: Introduction to Design Patterns (Singleton, Builder, Factory)	Design pattern implementation reflects real-world programming practices and software architecture strategies.
§130.280(c)(9)(A)	Identify and analyze career opportunities in computer programming.	Implied in certification pursuit and Objective 5 (Design Patterns for real-world application)	Preparation for the certification exam introduces learners to industry-aligned skills and credentials, supporting career readiness.
§130.280(c)(10)(A)	Identify and comply with legal and ethical responsibilities related to the use of technology.	4.5: Encapsulation; general understanding of professional programming practices	Encapsulation and access control mirror ethical programming practices and secure coding techniques.