**Document Generated: 04/06/2026**

**Learning Style: Virtual Classroom**

**Technology:**

**Difficulty: Beginner**

**Course Duration: 5 Days**

**Next Course Date: May 11, 2026**

# Getting Started with Programming in C# / .Net Core for Non-Developers (TTCN10975)

## About This Course:

Getting Started with Programming, OO and C# / .Net Core Basics for Non-Developers is a fast-paced, five day course designed to provide you with a first look

at how to code to a very basic level. You'll gain light hands-on programming experience, while you begin your journey to develop a programmer's mindset. This course aligns with the Microsoft Official Curriculum (MOC) 10975

Throughout the course you'll explore the intricacies of the application development cycle, program structure, and language syntax. You'll learn and practice core coding skills including fundamental OO concepts, vital programming constructs, string and character manipulation, dynamic memory allocation, standard input/output, and exception handling. This course quickly introduces developers to essential programming concepts as well as implementing software in C# using the .Net framework and Visual Studio. Students will learn about project organization, debugging, and programming. The focus of this course is on core programming concepts such as computer storage, data types, decision structures, and repetition by using loops. The course also covers an introduction to object-oriented programming covering classes, encapsulation, inheritance, and polymorphism. Coverage is also included around exception handling, application security, performance, and memory management.

## Course Objectives:

- Explain core programming fundamentals such as computer storage and processing.
- Explain computer number systems such as binary.
- Create and use variables and constants in programs.
- Explain how to create and use functions in a program.
- Create and use decisions structures in a computer program.
- Create and use repetition (loops) in a computer program.
- Explain pseudocode and its role in programming.
- Explain the basic computer data structures such as arrays, lists, stacks, and queues.
- Implement object-oriented programming concepts.
- Create and use classes in a computer program.
- Implement encapsulation, inheritance, and polymorphism.
- Describe the base class library (BCL) in the .NET Framework.
- Explain the application security concepts.
- Implement simple I/O in a computer program.

- Identify application errors and explain how to debug an application and handle errors.
- Identify the performance considerations for applications.
- Use Threat Modeling to identify potential vulnerabilities in a real-life case study

## Audience:

- Ability to use computers to start programs, open and save files, navigate application menus and interfaces
- Ability to understand logical concepts such as comparisons
- Understand number theory
- Ability to create, understand, and follow structured directions or step-by-step procedures
- Ability to understand and apply abstract concepts to concrete examples

## Prerequisites:

- Ability to use computers to start programs, open and save files, navigate application menus and interfaces
- Ability to understand logical concepts such as comparisons
- Understand number theory
- Ability to create, understand, and follow structured directions or step-by-step procedures
- Ability to understand and apply abstract concepts to concrete examples

## Course Outline:

Module 1: Introduction to Core Programming Concepts

This module provides background and foundational information on how computers process information, discusses the different types of applications that a programmer might be creating, and then provides information on how code is compiled and interpreted by a computer.

After completing this module, students will be able to:

- Describe computer data storage and processing concepts
- Describe application types
- Describe the lifecycle of an application
- Describe code compilation

Lessons

- Computer Data Storage and Processing
- Application Types
- Application Life-Cycle
- Code Compilation
- Lab : Thinking Like a Computer

Module 2: Core Programming Language Concepts

This module covers programming language syntax and the importance of using good syntax and following the syntax rules for the chosen language. This module also discusses the core data types and how to store these data types in computer memory by using variables and constants.

After completing this module, students will be able to:

- Define syntax
- Explain the different types of core data used in programs
- Declare and use variables and constants in a computer program

Lessons

- Syntax
- Data Types
- Variables and Constants
- Lab : Working with Data Types

Module 3: Program Flow

This module covers how code is executed in a computer program, such as top to bottom, in structured programming and branching in code execution. The module teaches these concepts through the use of functions, decision structures, and looping constructs.

After completing this module, students will be able to:

- Describe structured programming
- Create and use functions in your code
- Create and use decision structures
- Create and use looping structures

Lessons

- Introduction to Structured Programming Concepts
- Introduction to Branching
- Using Functions
- Using Decision Structures
- Introducing Repetition
- Lab : Creating Functions, Decisions, and Looping

Module 4: Algorithms and Data Structures

This module introduces the concept of an algorithm by examining a daily routine such as a morning routine for getting up and going to work, outlining all the steps required including the decisions to be made as the routine progresses. The module also discusses how to translate these set of steps into pseudo code for evaluation of the algorithm that will be translated into actual code.

After completing this module, you will be able to:

- Transfer problem statements into pseudo code
- Create algorithms
- Translate pseudo code into programming code
- Create simple algorithms in code

- Create data structures to store data

Lessons

- Understand How to Write Pseudo Code
- Algorithm Examples
- Introduction to Data Structures
- Lab : Working with Algorithms and Data Structures

Module 5: Error Handling and Debugging

This module helps students understand that errors are a part of programming and they must understand how to anticipate errors, handle those errors in code, and present a good user experience with a program. This module introduces structured exception handling as the mechanism to deal with errors. After completing this module, students will be able to:

- Implement structured exception handling
- Debug applications by using Visual Studio

Lessons

- Introduction to Program Errors
- Introduction to Structured Error Handling
- Introduction to Debugging in Visual Studio
- Lab : Implementing Debugging and Error Handling

Module 6: Introduction to Object-Oriented Programming

This module covers an introduction to the concepts related to object-oriented programming (OOP). The content has been split across two modules with this module focusing on basic OOP concepts that will provide sufficient knowledge to understand complex data structures starting with structs and then moving onto classes. This module helps the students gain an understanding of how to encapsulate data and related functionality within a class. After completing this module, students will be able to:

- Create and use structure types

- Create and use basic class files
- Choose when to use a struct vs a class

Lessons

- Introduction to Complex Structures
- Introduction to Structs
- Introduction to Classes
- Introducing Encapsulation
- Lab : Implementing Complex Data Structures

Module 7: More Object-Oriented Programming

This module teaches students about inheritance and polymorphism in classes and function overloading. Function overloading and polymorphism can go hand-in-hand as often times when you inherit from a class, you want to override or change the existing behavior to suit the needs of you class.

The module also provides an introduction to the base class library in the .NET Framework so that students can start to think about the existence of functionality in other class files and how they can search the .NET Framework to find this functionality and take advantage of it. After completing this module, students will be able to:

- Use inheritance in OOP
- Implement polymorphism in your classes
- Describe how the base class library is constructed
- Find class information by using the Object Browser

Lessons

- Introduction to Inheritance
- Introduction to Polymorphism
- Introduction to the .NET Framework and the Base Class Library
- Lab : Implementing Inheritance

- Lab : Implementing Polymorphism

Module 8: Core I/O Programming

This module introduces some core input/output (I/O) concepts that programmers will use while creating applications. Starting with console I/O, this module introduces input and output to the Console window.

The module also talks about reading and writing files, which is an important concept to know because applications work with the files on the disk systems on computers. After completing this module, students will be able to:

- Read input from a console
- Output data to the console
- Read and write text files

Lessons

- Using Console I/O
- Using File I/O
- Lab : Core I/O Programming

Module 9: Application Performance and Memory Management

This module enables students understand that memory on a computer is a finite resource. It talks about how good application design and good coding discipline with memory conservation and memory management will help programmers learn to develop applications that users will like. This is because these applications will be fast, responsive, and do not negatively impact other applications.

After completing this module, students will be able to:

- Implement value and reference types correctly in an application
- Convert between value types and reference types
- Use the garbage collector

Lessons

- Value Types vs Reference Types

- Converting Types
- The Garbage Collector
- Lab : Using Value Types and Reference Types