

This thorough and comprehensive 5-day course is a practical introduction to programming in C#, utilizing the services provided by .NET. This course emphasizes the C# language. It is current to Visual Studio 2015. Important newer features such as dynamic data type, named and optional arguments, the use of variance in generic interfaces, and asynchronous programming keywords are covered in a final chapter. A supplement covers the fundamentals of Language Integrated Query (LINQ). This course is intended to be fully accessible to programmers who do not already have a strong background in object-oriented programming in C-like languages, such as C++ or Java. It is ideal, for example, for Visual Basic 6 or COBOL programmers who desire to learn C#.

This course introduces object-oriented concepts early, and C# is developed in a way that leverages its object orientation. A case study is used to illustrate creating a complete system using C# and .NET. Besides supporting traditional object-oriented features, such as classes, inheritance, and polymorphism, C# introduces several additional features, such as properties, indexers, delegates, events, and interfaces that make C# a compelling language for developing object-oriented and component-based systems. This course provides thorough coverage of all these features.

Course Objectives:

- Acquire a working knowledge of C# programming
- Learn how to implement programs using C# and classes from the .NET Framework
- Learn how to implement simple GUI programs using Windows Forms
- Gain a working knowledge of dynamic data type, named and optional arguments, and other new features in C# 4.0.
- Learn how to do asynchronous programming using new keywords in C# 5.0.
- Become aware of new features in C# 6.0

Audience: Programmers who need to design and develop C# for the .NET framework.

Prerequisites: The student should have programming experience in a high-level language.

Number of Days: 5 days

1 .NET: What You Need to Know Getting Started .NET: What is <i>Really</i> Happening .NET Programming in a Nutshell Viewing the Assembly Viewing Intermediate Language Understanding .NET Visual Studio 2013 Creating a Console Application Adding a C# file	Using the Visual Studio Text Editor IntelliSense Build and Run the Project Pausing the Output Visual C# and GUI Programs .NET Documentation
2 First C# Programs Hello, World	

	Compiling, Running (Command Line)		Increment and Decrement
	Program Structure		Relational Operators
	Namespaces		Conditional Logical Operators
	Exercise		Short-Circuit Evaluation
	Answer		Ternary Conditional Operators
	Variables		Bitwise Operators
	Expressions		Bitwise Logical Operators
	Assignment		Bitwise Shift Operators
	Calculations Using C#		Assignment Operators
	More about Output in C#		Expressions
	Input in C#		Precedence
	More about Classes		Associativity
	InputWrapper Class		Checking
	Echo Program	5	Control Structures
	Using InputWrapper		If Test
	Compiling Multiple Files		Blocks
	Multiple Files in Visual Studio		Loops
	The .NET Framework		while Loop
3	Data Types in C#		do while Loops
	Strong Typing		for Loops
	Typing in C#		Arrays
	Typing in C++		foreach Loop
	Typing in Visual Basic 6		break
	C# Types		continue
	Integer Types		goto
	Integer Type Range		Structure Programming
	Integer Literals		Multiple Methods
	Floating Point Types		switch
	Floating Point Literals		switch in C# and C/C++
	IEEE Standard for Floating Point	6	Object-Oriented Programming
	Decimal Type		Objects
	Decimal Literals		Objects in the Real World
	Character Type		Object Models
	Character Literals		Reusable Software Components
	string		Objects in Software
	Escape Characters		State and Behavior
	Boolean Type		Abstraction
	Implicit Conversions		Encapsulation
	Explicit Conversions		Classes
	Nullable Types		Inheritance Concept
4.	Operators and Expressions		Relationships among Classes
	Operator Cardinality		Polymorphism
	Arithmetic Operators		Object-Oriented Analysis and Design
	Multiplication		Use Cases
	Division		CRC Cards and UML
	Additive Operators		

7	<p>Classes</p> <ul style="list-style-type: none"> Classes as Structure Data Classes and Objects References Instantiating and Using an Object Assigning Object References Garbage Collection Methods Public and Private Abstraction Encapsulation Initialization Initialization with Constructors Default Constructor this Static Fields and Methods Static Methods Static Constructor Constant and Readonly Fields 		<ul style="list-style-type: none"> Structure Parameters Class parameters Method Overloading Modifiers as Part of the Signature Variable Length Parameter Lists Properties Auto-Implemented Properties Operator Overloading Operator Overloading in the Class Library
8	<p>More about Types</p> <ul style="list-style-type: none"> Overview of Types in C# Structures Uninitialized Variables Copying a Structure Hotel.cs HotelCopy.cs Classes and Structs Enumeration Types Reference Types Class Types object string Arrays Default Values Boxing and Unboxing Implicitly Types Variables 	10	<p>Characters and Strings</p> <ul style="list-style-type: none"> Characters Character Codes ASCII and Unicode Escape Sequences Strings String Class String Literals and Initialization Concatenation Index Relational Operators String Equality String Comparisons String Input String Methods and Properties StringBuilder Class StringBuilder Equality Command Line Arguments Command Line Arguments in the IDE Command Loops Splitting a String
9	<p>Methods, Properties, and Operators</p> <ul style="list-style-type: none"> Static and Instance Methods Method Parameters No “Freestanding” Functions in C# Classes with All Static Methods Parameter Passing Parameter Terminology Value Parameters Reference Parameters Output Parameters 	11	<p>Arrays and Indexers</p> <ul style="list-style-type: none"> Arrays One Dimensional Arrays System.Array Random Number Generation Next Methods Jagged Arrays Rectangular Arrays Arrays as Collections Account Class Bank Class TestBank Class ATM Class

	Running the Case Study		Account
	Indexers		CheckingAccount,
	Using the Indexer		SavingsAccount
12	Inheritance		Bank and ATM
	Inheritance Fundamentals		TestBank
	Inheritance in C#	14	Formatting and Conversion
	Single Inheritance		Introduction to Formatting
	Root Class – <i>Object</i>		ToString
	Access Control		ToString in Your Own Class
	Public Class Accessibility		Using Placeholders
	Internal Class Accessibility		Format Strings
	Member Accessibility		Simple Placeholders
	Member Accessibility Qualifiers		Controlling Width
	Method Hiding		Format String
	Method Hiding and Overriding		Currency
	Initialization		String.Format
	Initialization Fundamentals		PadLeft and PadRight
	Default Constructor		Type Conversions
	Overloaded Constructors		Conversion of Built-In Types
	Invoking Base Class Constructors		Conversion of User-Defined
	Bank Case Study Analysis		Types
	Account	15	Exceptions
	CheckingAccount		Introduction to Exceptions
	SavingsAccount		Exception Fundamentals
	TestAccount		.NET Exception Handling
	Running the Case Study		Exception Flow of Control
13	Virtual Methods and Polymorphism		Context and Stack Unwinding
	Introduction to Polymorphism		System.Exception
	Abstract and Sealed Classes		User-Defined Exception Classes
	Virtual Methods and Dynamic Binding		Structure Exception Handling
	Type Conversions in Inheritance		Finally Block
	Converting Down the Hierarchy		Inner Exceptions
	Converting Up the Hierarchy		Checked Integer Arithmetic
	Virtual Methods	16.	Interfaces
	Virtual Method Cost		Interfaces in C#
	Method overriding		Interface Inheritance
	The Fragile Base Class Problem		Programming with Interfaces
	<i>override</i> Keyword		Implementing Interfaces
	Polymorphism		Using an Interface
	Polymorphism Using “Type Tags”		Dynamic Use of Interfaces
	Polymorphism Using Virtual		is Operator
	Abstract Classes		as Operator
	Sealed Classes		Common Interfaces in Case
	Heterogeneous Collections		Study – IAccount
	Case Study Classes		Apparent Redundancy
	Run the Case Study		IStatement

<ul style="list-style-type: none"> IStatement Methods IChecking ISavings The Implementation SavingsAccount The Client Resolving Ambiguity Access Modifier Explicit Interfaces Test Program 	<ul style="list-style-type: none"> 17 	<ul style="list-style-type: none"> .NET Interfaces and Collections Collections Count and Capacity foreach Loop Array Notation Adding to the List Remove Method RemoveAt Method Collection Interfaces IEnumerable and IEnumerator ICollection ICollection IList A Collection of User-Defined Objects Duplicate Objects A Correction to AccountList (Step 1) Copy Semantics and ICloneable Copy Semantics in C# Shallow Copy and Deep Copy Reference Copy Memberwise Clone Using ICloneable Comparing Objects Sorting an Array Anatomy of Array.Sort Using the is Operator The Use of Dynamic Type Checking Implementing IComparable Running the Program Complete Solution Writing Generic Code Using a Class of <i>object</i> Generic Types Generic Syntax in C# Generic Client Code System.Collections.Generic Object Initializers Collection Initializers 	<ul style="list-style-type: none"> 18 	<ul style="list-style-type: none"> Anonymous Types Delegates and Events Overview of Delegates and Events Callbacks and Delegates Usage of Delegates Declaring a Delegate Defining a Method Creating a Delegate Object Calling a Delegate A Random Array Anonymous Methods Combining Delegate Objects Account.cs DelegateAccount.cs Lambda Expressions Named Method Anonymous Method Events Events in C# and .NET Client Side Event Code 	<ul style="list-style-type: none"> 19 	<ul style="list-style-type: none"> Introduction to Windows Forms Creating a Windows Forms App Partial Classes Windows Forms Event Handling Add Events for a Control Events Documentation Closing a Form Listbox Control 	<ul style="list-style-type: none"> 20 	<ul style="list-style-type: none"> New Features in C# <i>dynamic</i> Type <i>dynamic</i> versus <i>object</i> Behavior of <i>object</i> Behavior of <i>dynamic</i> Names Arguments Optional Arguments Book Class Using Optional Arguments Variance in Generic Interfaces Variance with IComparer<T> Interfaces with Variance Support Asynchronous Programs in C# 5.0 Task and Task<TResult> Aysnc Methods
---	--	---	--	--	--	--	--	---

Synchronous Call

Async Call

Threading

New Features in C# 6.0

Null-Conditional Operator

21 Appendix A – Learning Resources