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 Really 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

2. First C# Programs
   - Hello, World

Using the Visual Studio Text Editor
IntelliSense
Build and Run the Project
Pausing the Output
Visual C# and GUI Programs
.NET Documentation

© Batky-Howell, LLC
Object-Oriented Programming in C# (VS 2015)
Compiling, Running (Command Line)
Program Structure
Namespaces
Exercise
Answer
Variables
Expressions
Assignment
Calculations Using C#
More about Output in C#
Input in C#
More about Classes
InputWrapper Class
Echo Program
Using InputWrapper
Compiling Multiple Files
Multiple Files in Visual Studio
The .NET Framework

3 Data Types in C#
Strong Typing
Typing in C#
Typing in C++
Typing in Visual Basic 6
C# Types
Integer Types
Integer Type Range
Integer Literals
Floating Point Types
Floating Point Literals
IEEE Standard for Floating Point
Decimal Type
Decimal Literals
Character Type
Character Literals
string
Escape Characters
Boolean Type
Implicit Conversions
Explicit Conversions
Nullable Types

4 Operators and Expressions
Operator Cardinality
Arithmetic Operators
Multiplication
Division
Additive Operators
Increment and Decrement
Relational Operators
Conditional Logical Operators
Short-Circuit Evaluation
Ternary Conditional Operators
Bitwise Operators
Bitwise Logical Operators
Bitwise Shift Operators
Assignment Operators
Expressions
Precedence
Associativity
Checking

5 Control Structures
If Test
Blocks
Loops
while Loop
do while Loops
for Loops
Arrays
foreach Loop
break
continue
goto
Structure Programming
Multiple Methods
switch
switch in C# and C/C++

6 Object-Oriented Programming
Objects
Objects in the Real World
Object Models
Reusable Software Components
Objects in Software
State and Behavior
Abstraction
Encapsulation
Classes
Inheritance Concept
Relationships among Classes
Polymorphism
Object-Oriented Analysis and Design
Use Cases
CRC Cards and UML
7 Classes
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

8 More about Types
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

9 Methods, Properties, and Operators
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

10 Characters and Strings
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

11 Arrays and Indexers
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
Indexers
Using the Indexer

12 Inheritance
Inheritance Fundamentals
Inheritance in C#
Single Inheritance
Root Class – Object
Access Control
Public Class Accessibility
Internal Class Accessibility
Member Accessibility
Member Accessibility Qualifiers
Method Hiding
Method Hiding and Overriding
Initialization
Initialization Fundamentals
Default Constructor
Overloaded Constructors
Invoking Base Class Constructors
Bank Case Study Analysis
Account
CheckingAccount,
SavingsAccount
Bank and ATM
TestBank

14 Formatting and Conversion
Introduction to Formatting
ToString
ToString in Your Own Class
Using Placeholders
Format Strings
Simple Placeholders
Controlling Width
Format String
Currency
String.Format
PadLeft and PadRight
Type Conversions
Conversion of Built-In Types
Conversion of User-Defined Types

15 Exceptions
Introduction to Exceptions
Exception Fundamentals
.NET Exception Handling
Exception Flow of Control
Context and Stack Unwinding
System.Exception
User-Defined Exception Classes
Structure Exception Handling
Finally Block
Inner Exceptions
Checked Integer Arithmetic

16 Interfaces
Interfaces in C#
Interface Inheritance
Programming with Interfaces
Implementing Interfaces
Using an Interface
Dynamic Use of Interfaces
is Operator
as Operator
Common Interfaces in Case Study – IAccount
Apparent Redundancy
IStatement
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>.NET Interfaces and Collections</td>
</tr>
<tr>
<td>18</td>
<td>Anonymous Types</td>
</tr>
<tr>
<td>19</td>
<td>Introduction to Windows Forms</td>
</tr>
<tr>
<td>20</td>
<td>New Features in C#</td>
</tr>
</tbody>
</table>

**IStatement Methods**
- IChecking
- ISavings

**The Implementation**
- SavingsAccount
- The Client
- Resolving Ambiguity
- Access Modifier
- Explicit Interfaces Test Program

**.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
  - 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 object
- Generic Types
- Generic Syntax in C#
- Generic Client Code
- System.Collections.Generic
- Object Initializers
- Collection Initializers

**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

**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

**New Features in C#**
- dynamic Type
  - dynamic versus object
- Behavior of object
  - Behavior of dynamic
- 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