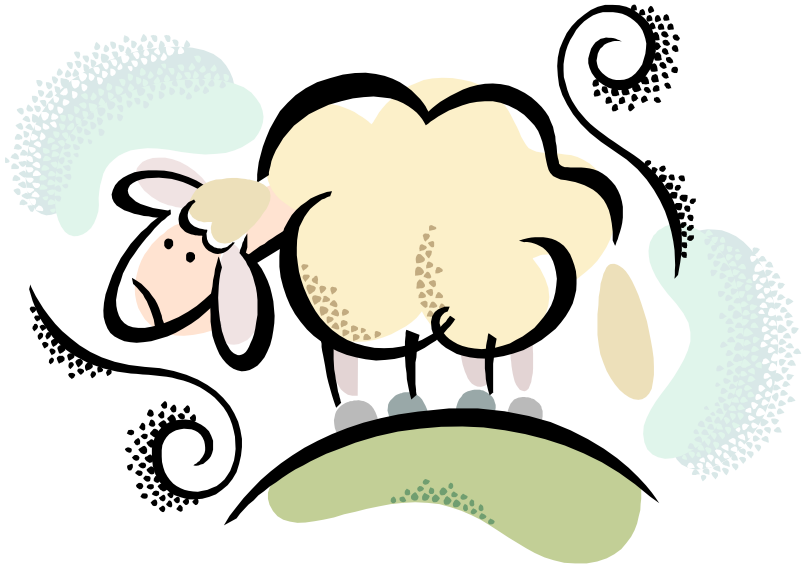

C# Programming IV-7:

System.Numerics Namespace

Legal Stuff

This presentation as well as other presentations and documents found on the sheepsqueezers.com website may contain quoted material from outside sources such as books, articles and websites. It is our intention to diligently reference all outside sources. Occasionally, though, a reference may be missed. No copyright infringement whatsoever is intended, and all outside source materials are copyright of their respective author(s).

.NET Lecture Series

	C# Programming I: Concepts of OOP	*C# Programming II: Beginning C#*	*C# Programming III: Advanced C#*	
C# Programming IV-1: System Namespace	*C# Programming IV-2: System.Collections Namespace*	*C# Programming IV-3: System.Collections. Generic Namespace*	*C# Programming IV-4A: System.Data Namespace*	*C# Programming IV-4B: System.Data.Odbc Namespace*
C# Programming IV-4C: System.Data.OleDb Namespace	*C# Programming IV-4D: Oracle.DataAccess.Client Namespace*	*C# Programming IV-4E: System.Data.SqlClient Namespace*	*C# Programming IV-4F: System.Data.SqlTypes Namespace*	*C# Programming IV-5: System.Drawing/(2D) Namespace*
C# Programming IV-6: System.IO Namespace	*C# Programming IV-7: System.Numerics*	*C# Programming IV-8: System.Text and System.Text. RegularExpressions Namespaces*	*C# Programming V: Introduction to LINQ*	

C# Self-Inflicted Project #1

Address Cleaning

C# Self-Inflicted Project #2

Large Intersection Problem

Charting Our Course

- The `System.Numerics` Namespace
- What Next?

The `System.Numerics` Namespace

The `System.Numerics` namespace is defined by Microsoft as follows:

The System.Numerics namespace contains types that allow reading and writing to files and data streams, and types that provide basic file and directory support.

Note that `System.Numerics` was introduced in .NET Framework 4.0!!

When writing code using this namespace, include the following line at the top of your C# program:

```
using System.Numerics;
```

The `System.Numerics` Namespace

- → Classes
  - → `DeflateStream`

Classes

There are no classes in this namespace. See the structures section for more.

The `System.Numerics` Namespace

→ Attributes

Attributes

There are no attributes in the `System.Numerics` namespace.

The `System.Numerics` Namespace

→ EventArgs

EventArgs

There are no `EventArgs` in the `System.Numerics` namespace.

The `System.Numerics` Namespace

→ Structures

Structures

Below are the structures available in the `System.Numerics` namespace.

Structures

- BigInteger – Represents an arbitrarily large signed integer.
- Complex – Represents a complex number.

The `BigInteger` structure Represents an arbitrarily large signed integer. According to Microsoft's website: *The BigInteger type is an immutable type that represents an arbitrarily large integer whose value in theory has no upper or lower bounds. The members of the BigInteger type closely parallel those of other integral types (the Byte, Int16, Int32, Int64, SByte, UInt16, UInt32, and UInt64 types). This type differs from the other integral types in the .NET Framework, which have a range indicated by their MinValue and MaxValue properties.* Below are the members of this structure.

Constructors

- BigInteger(Byte[]) - Initializes a new instance of the BigInteger structure using the values in a byte array
- BigInteger(Decimal) - Initializes a new instance of the BigInteger structure using a Decimal value
- BigInteger(Double) - Initializes a new instance of the BigInteger structure using a double-precision floating-point value
- BigInteger(Int32) - Initializes a new instance of the BigInteger structure using a 32-bit signed integer value
- BigInteger(Int64) - Initializes a new instance of the BigInteger structure using a 64-bit signed integer value
- BigInteger(Single) - Initializes a new instance of the BigInteger structure using a single-precision floating-point value
- BigInteger(UInt32) - Initializes a new instance of the BigInteger structure using an unsigned 32-bit integer value
- BigInteger(UInt64) - Initializes a new instance of the BigInteger structure with an unsigned 64-bit integer value

The `System.Numerics` Namespace

→ Structures

Structures

Below are the structures available in the `System.Numerics` namespace.

Structures

- BigInteger – Represents an arbitrarily large signed integer.
- Complex – Represents a complex number.

The `BigInteger` structure represents an arbitrarily large signed integer. According to Microsoft's website: *The BigInteger type is an immutable type that represents an arbitrarily large integer whose value in theory has no upper or lower bounds. The members of the BigInteger type closely parallel those of other integral types (the Byte, Int16, Int32, Int64, SByte, UInt16, UInt32, and UInt64 types). This type differs from the other integral types in the .NET Framework, which have a range indicated by their MinValue and MaxValue properties.* Below are the members of this structure.

Constructors

- BigInteger(Byte[]) - Initializes a new instance of the BigInteger structure using the values in a byte array
- BigInteger(Decimal) - Initializes a new instance of the BigInteger structure using a Decimal value
- BigInteger(Double) - Initializes a new instance of the BigInteger structure using a double-precision floating-point value
- BigInteger(Int32) - Initializes a new instance of the BigInteger structure using a 32-bit signed integer value
- BigInteger(Int64) - Initializes a new instance of the BigInteger structure using a 64-bit signed integer value
- BigInteger(Single) - Initializes a new instance of the BigInteger structure using a single-precision floating-point value
- BigInteger(UInt32) - Initializes a new instance of the BigInteger structure using an unsigned 32-bit integer value
- BigInteger(UInt64) - Initializes a new instance of the BigInteger structure with an unsigned 64-bit integer value

Structures

Properties

- IsEven - Indicates whether the value of the current BigInteger object is an even number
- IsOne - Indicates whether the value of the current BigInteger object is BigInteger.One
- IsPowerOfTwo - Indicates whether the value of the current BigInteger object is a power of two
- IsZero - Indicates whether the value of the current BigInteger object is BigInteger.Zero
- MinusOne - Gets a value that represents the number negative one (-1)
- One - Gets a value that represents the number one (1)
- Sign - Gets a number that indicates the sign (negative, positive, or zero) of the current BigInteger object
- Zero - Gets a value that represents the number 0 (zero)

Methods

- Abs - Gets the absolute value of a BigInteger object
- Add - Adds two BigInteger values and returns the result
- Compare - Compares two BigInteger values and returns an integer that indicates whether the first value is less than, equal to, or greater than the second value
- CompareTo(BigInteger) - Compares this instance to a second BigInteger and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the specified object
- CompareTo(Int64) - Compares this instance to a signed 64-bit integer and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the signed 64-bit integer
- CompareTo(Object) - Compares this instance to a specified object and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the specified object
- CompareTo(UInt64) - Compares this instance to an unsigned 64-bit integer and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the unsigned 64-bit integer
- Divide - Divides one BigInteger value by another and returns the result
- DivRem - Divides one BigInteger value by another, returns the result, and returns the remainder in an output parameter
- Equals(BigInteger) - Returns a value that indicates whether the current instance and a specified BigInteger object have the same value
- Equals(Int64) - Returns a value that indicates whether the current instance and a signed 64-bit integer have the same value
- Equals(Object) Returns a value that indicates whether the current instance and a specified object have the same value. (Overrides ValueType.Equals(Object).)
- Equals(UInt64) - Returns a value that indicates whether the current instance and an unsigned 64-bit integer have the same value
- Finalize Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode Returns the hash code for the current BigInteger object. (Overrides ValueType.GetHashCode().)
- GetType Gets the Type of the current instance. (Inherited from Object.)
- GreatestCommonDivisor - Finds the greatest common divisor of two BigInteger values
- Log(BigInteger) - Returns the natural (base e) logarithm of a specified number
- Log(BigInteger, Double) - Returns the logarithm of a specified number in a specified base
- Log10 - Returns the base 10 logarithm of a specified number

Structures

Methods (continued)

- Max - Returns the larger of two BigInteger values
- MemberwiseClone Creates a shallow copy of the current Object. (Inherited from Object.)
- Min - Returns the smaller of two BigInteger values
- ModPow - Performs modulus division on a number raised to the power of another number
- Multiply - Returns the product of two BigInteger values
- Negate - Negates a specified BigInteger value
- Parse(String) - Converts the string representation of a number to its BigInteger equivalent
- Parse(String, NumberStyles) - Converts the string representation of a number in a specified style to its BigInteger equivalent
- Parse(String, IFormatProvider) - Converts the string representation of a number in a specified culture-specific format to its BigInteger equivalent
- Parse(String, NumberStyles, IFormatProvider) - Converts the string representation of a number in a specified style and culture-specific format to its BigInteger equivalent
- Pow - Raises a BigInteger value to the power of a specified value
- Remainder - Performs integer division on two BigInteger values and returns the remainder
- Subtract - Subtracts one BigInteger value from another and returns the result
- ToByteArray - Converts a BigInteger value to a byte array
- ToString() Converts the numeric value of the current BigInteger object to its equivalent string representation. (Overrides ValueType.ToString().)
- ToString(IFormatProvider) - Converts the numeric value of the current BigInteger object to its equivalent string representation by using the specified culture-specific formatting information
- ToString(String) - Converts the numeric value of the current BigInteger object to its equivalent string representation by using the specified format
- ToString(String, IFormatProvider) - Converts the numeric value of the current BigInteger object to its equivalent string representation by using the specified format and culture-specific format information
- TryParse(String, BigInteger) - Tries to convert the string representation of a number to its BigInteger equivalent, and returns a value that indicates whether the conversion succeeded
- TryParse(String, NumberStyles, IFormatProvider, BigInteger) - Tries to convert the string representation of a number in a specified style and culture-specific format to its BigInteger equivalent, and returns a value that indicates whether the conversion succeeded

Operators

- Addition - Adds the values of two specified BigInteger objects
- BitwiseAnd - Performs a bitwise And operation on two BigInteger values
- BitwiseOr - Performs a bitwise Or operation on two BigInteger values
- Decrement - Decrements a BigInteger value by 1
- Division - Divides a specified BigInteger value by another specified BigInteger value by using integer division

Structures

Operators (continued)

- Equality(BigInteger, Int64) - Returns a value that indicates whether a BigInteger value and a signed long integer value are equal
- Equality(BigInteger, BigInteger) - Returns a value that indicates whether the values of two BigInteger objects are equal
- Equality(BigInteger, UInt64) - Returns a value that indicates whether a BigInteger value and an unsigned long integer value are equal
- Equality(Int64, BigInteger) - Returns a value that indicates whether a signed long integer value and a BigInteger value are equal
- Equality(UInt64, BigInteger) - Returns a value that indicates whether an unsigned long integer value and a BigInteger value are equal
- ExclusiveOr - Performs a bitwise exclusive Or (XOr) operation on two BigInteger values
- Explicit(BigInteger to Byte) - Defines an explicit conversion of a BigInteger object to an unsigned byte value
- Explicit(BigInteger to Int32) - Defines an explicit conversion of a BigInteger object to a 32-bit signed integer value
- Explicit(BigInteger to UInt64) - Defines an explicit conversion of a BigInteger object to an unsigned 64-bit integer value
- Explicit(BigInteger to Single) - Defines an explicit conversion of a BigInteger object to a single-precision floating-point value
- Explicit(BigInteger to Double) - Defines an explicit conversion of a BigInteger object to a Double value
- Explicit(BigInteger to Int64) - Defines an explicit conversion of a BigInteger object to a 64-bit signed integer value
- Explicit(BigInteger to Int16) - Defines an explicit conversion of a BigInteger object to a 16-bit signed integer value
- Explicit(BigInteger to SByte) - Defines an explicit conversion of a BigInteger object to a signed 8-bit value
- Explicit(BigInteger to UInt32) - Defines an explicit conversion of a BigInteger object to an unsigned 32-bit integer value
- Explicit(BigInteger to UInt16) - Defines an explicit conversion of a BigInteger object to an unsigned 16-bit integer value
- Explicit(BigInteger to Decimal) - Defines an explicit conversion of a BigInteger object to a Decimal value
- Explicit(Decimal to BigInteger) - Defines an explicit conversion of a Decimal object to a BigInteger value
- Explicit(Double to BigInteger) - Defines an explicit conversion of a Double value to a BigInteger value
- Explicit(Single to BigInteger) - Defines an explicit conversion of a Single object to a BigInteger value
- GreaterThan(BigInteger, Int64) - Returns a value that indicates whether a BigInteger is greater than a 64-bit signed integer value
- GreaterThan(BigInteger, BigInteger) - Returns a value that indicates whether a BigInteger value is greater than another BigInteger value
- GreaterThan(BigInteger, UInt64) - Returns a value that indicates whether a BigInteger value is greater than a 64-bit unsigned integer
- GreaterThan(Int64, BigInteger) - Returns a value that indicates whether a 64-bit signed integer is greater than a BigInteger value
- GreaterThan(UInt64, BigInteger) - Returns a value that indicates whether a BigInteger value is greater than a 64-bit unsigned integer
- GreaterThanOrEqual(BigInteger, Int64) - Returns a value that indicates whether a BigInteger value is greater than or equal to a 64-bit signed integer value
- GreaterThanOrEqual(BigInteger, BigInteger) - Returns a value that indicates whether a BigInteger value is greater than or equal to another BigInteger value
- GreaterThanOrEqual(BigInteger, UInt64) - Returns a value that indicates whether a BigInteger value is greater than or equal to a 64-bit unsigned integer value
- GreaterThanOrEqual(Int64, BigInteger) - Returns a value that indicates whether a 64-bit signed integer is greater than or equal to a BigInteger value

Structures

Operators (continued)

- GreaterThanOrEqual(Int64, BigInteger) - Returns a value that indicates whether a 64-bit signed integer is greater than or equal to a BigInteger value
- GreaterThanOrEqual(UInt64, BigInteger) - Returns a value that indicates whether a 64-bit unsigned integer is greater than or equal to a BigInteger value
- Implicit(Byte to BigInteger) - Defines an implicit conversion of an unsigned byte to a BigInteger value
- Implicit(Int16 to BigInteger) - Defines an implicit conversion of a signed 16-bit integer to a BigInteger value
- Implicit(Int32 to BigInteger) - Defines an implicit conversion of a signed 32-bit integer to a BigInteger value
- Implicit(Int64 to BigInteger) - Defines an implicit conversion of a signed 64-bit integer to a BigInteger value
- Implicit(SByte to BigInteger) - Defines an implicit conversion of an 8-bit signed integer to a BigInteger value
- Implicit(UInt16 to BigInteger) - Defines an implicit conversion of a 16-bit unsigned integer to a BigInteger value
- Implicit(UInt32 to BigInteger) - Defines an implicit conversion of a 32-bit unsigned integer to a BigInteger value
- Implicit(UInt64 to BigInteger) - Defines an implicit conversion of a 64-bit unsigned integer to a BigInteger value
- Increment - Increments a BigInteger value by 1
- Inequality(BigInteger, Int64) - Returns a value that indicates whether a BigInteger value and a 64-bit signed integer are not equal
- Inequality(BigInteger, BigInteger) - Returns a value that indicates whether two BigInteger objects have different values
- Inequality(BigInteger, UInt64) - Returns a value that indicates whether a BigInteger value and a 64-bit unsigned integer are not equal
- Inequality(Int64, BigInteger) - Returns a value that indicates whether a 64-bit signed integer and a BigInteger value are not equal
- Inequality(UInt64, BigInteger) - Returns a value that indicates whether a 64-bit unsigned integer and a BigInteger value are not equal
- LeftShift - Shifts a BigInteger value a specified number of bits to the left
- LessThan(BigInteger, Int64) - Returns a value that indicates whether a BigInteger value is less than a 64-bit signed integer
- LessThan(BigInteger, BigInteger) - Returns a value that indicates whether a BigInteger value is less than another BigInteger value
- LessThan(BigInteger, UInt64) - Returns a value that indicates whether a BigInteger value is less than a 64-bit unsigned integer
- LessThan(Int64, BigInteger) - Returns a value that indicates whether a 64-bit signed integer is less than a BigInteger value
- LessThan(UInt64, BigInteger) - Returns a value that indicates whether a 64-bit unsigned integer is less than a BigInteger value
- LessThanOrEqual(BigInteger, Int64) - Returns a value that indicates whether a BigInteger value is less than or equal to a 64-bit signed integer
- LessThanOrEqual(BigInteger, BigInteger) - Returns a value that indicates whether a BigInteger value is less than or equal to another BigInteger value
- LessThanOrEqual(BigInteger, UInt64) - Returns a value that indicates whether a BigInteger value is less than or equal to a 64-bit unsigned integer
- LessThanOrEqual(Int64, BigInteger) - Returns a value that indicates whether a 64-bit signed integer is less than or equal to a BigInteger value
- LessThanOrEqual(UInt64, BigInteger) - Returns a value that indicates whether a 64-bit unsigned integer is less than or equal to a BigInteger value

Structures

Operators (continued)

- Modulus - Returns the remainder that results from division with two specified BigInteger values
- Multiply - Multiplies two specified BigInteger values
- OnesComplement - Returns the bitwise one's complement of a BigInteger value
- RightShift - Shifts a BigInteger value a specified number of bits to the right
- Subtraction - Subtracts a BigInteger value from another BigInteger value
- UnaryNegation - Negates a specified BigInteger value
- UnaryPlus - Returns the value of the BigInteger operand. (The sign of the operand is unchanged.)

The `Complex` structure represents a complex number. There members of this structure are listed below.

Constructors

- Complex - Initializes a new instance of the Complex structure using the specified real and imaginary values

Properties

- Imaginary - Gets the imaginary component of the current Complex object
- Magnitude - Gets the magnitude (or absolute value) of a complex number
- Phase - Gets the phase of a complex number
- Real - Gets the real component of the current Complex object

Methods

- Abs - Gets the absolute value (or magnitude) of a complex number
- Acos - Returns the angle that is the arc cosine of the specified complex number
- Add - Adds two complex numbers and returns the result
- Asin - Returns the angle that is the arc sine of the specified complex number
- Atan - Returns the angle that is the arc tangent of the specified complex number
- Conjugate - Computes the conjugate of a complex number and returns the result
- Cos - Returns the cosine of the specified complex number
- Cosh - Returns the hyperbolic cosine of the specified complex number
- Divide - Divides one complex number by another and returns the result
- Equals(Complex) - Returns a value that indicates whether the current instance and a specified complex number have the same value
- Equals(Object) Returns a value that indicates whether the current instance and a specified object have the same value. (Overrides ValueType.Equals(Object).)
- Exp - Returns e raised to the power specified by a complex number

Structures

Methods (continued)

- Finalize Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- FromPolarCoordinates - Creates a complex number from a point's polar coordinates
- GetHashCode Returns the hash code for the current Complex object. (Overrides ValueType.GetHashCode().)
- GetType Gets the Type of the current instance. (Inherited from Object.)
- Log(Complex) - Returns the natural (base e) logarithm of a specified complex number
- Log(Complex, Double) - Returns the logarithm of a specified complex number in a specified base
- Log10 - Returns the base-10 logarithm of a specified complex number
- MemberwiseClone Creates a shallow copy of the current Object. (Inherited from Object.)
- Multiply - Returns the product of two complex numbers
- Negate - Returns the additive inverse of a specified complex number
- Pow(Complex, Double) - Returns a specified complex number raised to a power specified by a double-precision floating-point number
- Pow(Complex, Complex) - Returns a specified complex number raised to a power specified by a complex number
- Reciprocal - Returns the multiplicative inverse of a complex number
- Sin - Returns the sine of the specified complex number
- Sinh - Returns the hyperbolic sine of the specified complex number
- Sqrt - Returns the square root of a specified complex number
- Subtract - Subtracts one complex number from another and returns the result
- Tan - Returns the tangent of the specified complex number
- Tanh - Returns the hyperbolic tangent of the specified complex number
- ToString() Converts the value of the current complex number to its equivalent string representation in Cartesian form. (Overrides ValueType.ToString().)
- ToString(IFormatProvider) - Converts the value of the current complex number to its equivalent string representation in Cartesian form by using the specified culture-specific formatting information
- ToString(String) - Converts the value of the current complex number to its equivalent string representation in Cartesian form by using the specified format for its real and imaginary parts
- ToString(String, IFormatProvider) - Converts the value of the current complex number to its equivalent string representation in Cartesian form by using the specified format and culture-specific format information for its real and imaginary parts

Operators

- Addition - Adds two complex numbers
- Division - Divides a specified complex number by another specified complex number
- Equality - Returns a value that indicates whether two complex numbers are equal
- Explicit(BigInteger to Complex) - Defines an explicit conversion of a BigInteger value to a complex number
- Explicit(Decimal to Complex) - Defines an explicit conversion of a Decimal value to a complex number
- Implicit(Byte to Complex) - Defines an implicit conversion of an unsigned byte to a complex number
- Implicit(Double to Complex) - Defines an implicit conversion of a double-precision floating-point number to a complex number
- Implicit(Int16 to Complex) - Defines an implicit conversion of a 16-bit signed integer to a complex number

Structures

Operators (continued)

- Implicit(Int32 to Complex) - Defines an implicit conversion of a 32-bit signed integer to a complex number
- Implicit(Int64 to Complex) - Defines an implicit conversion of a 64-bit signed integer to a complex number
- Implicit(SByte to Complex) - Defines an implicit conversion of a signed byte to a complex number
- Implicit(Single to Complex) - Defines an implicit conversion of a single-precision floating-point number to a complex number
- Implicit(UInt16 to Complex) - Defines an implicit conversion of a 16-bit unsigned integer to a complex number
- Implicit(UInt32 to Complex) - Defines an implicit conversion of a 32-bit unsigned integer to a complex number
- Implicit(UInt64 to Complex) - Defines an implicit conversion of a 64-bit unsigned integer to a complex number
- Inequality - Returns a value that indicates whether two complex numbers are not equal
- Multiply - Multiplies two specified complex numbers
- Subtraction - Subtracts a complex number from another complex number
- UnaryNegation - Returns the additive inverse of a specified complex number

Fields

- ImaginaryOne - Returns a new Complex instance with a real number equal to zero and an imaginary number equal to one
- One - Returns a new Complex instance with a real number equal to one and an imaginary number equal to zero
- Zero - Returns a new Complex instance with a real number equal to zero and an imaginary number equal to zero

The `System.Numerics` Namespace

→ Interfaces

Interfaces

There are no interfaces in the `System.Numerics` namespace.

The `System.Numerics` Namespace

→ Delegates

Delegates

There are no delegates in the `System.Numerics` namespace.

The `System.Numeric` Namespaces

→ Enumerations

Enumerations

There are no enumerations available in the `System.Numerics` namespace.

The `System.Numerics` Namespace

→ Exceptions

Exceptions

There are no exceptions available in the `System.Numerics` namespace.

What Next?

In *C# Programming IV-#*, we look at specific classes within specific namespaces such as the System namespace, the System.Numerics namespace, etc.

References

Click the book titles below to read more about these books on Amazon.com's website.

- Introducing Microsoft LINQ, Paolo Pialorsi and Marco Russo, Microsoft Press, ISBN:9780735623910
- LINQ Pocket Reference, Joseph Albahari and Ben Albahari, O'Reilly Press, ISBN:9780596519247
- Inside C#, Tom Archer and Andrew Whitechapel, Microsoft Press, ISBN:0735616485
- C# 4.0 In a Nutshell, O'Reilly Press, Joseph Albahari and Ben Albahari, ISBN:9780596800956
- The Object Primer, Scott W. Ambler, Cambridge Press, ISBN:0521540186
- CLR via C#, Jeffrey Richter, Microsoft Press, ISBN:9780735621633

Support sheepsqueezers.com

If you found this information helpful, please consider supporting sheepsqueezers.com. There are several ways to support our site:

- ☐ Buy me a cup of coffee by clicking on the following link and donate to my PayPal account: Buy Me A Cup Of Coffee?.
- ☐ Visit my Amazon.com Wish list at the following link and purchase an item: http://amzn.com/w/3OBK1K4EIWIR6

Please let me know if this document was useful by e-mailing me at `comments@sheepsqueezers.com`.