

## Anhang B

# Syntaxvergleich: C# 4.0 (C# 2010) versus Visual Basic 10.0 (VB 2010)

### In diesem Anhang:

Befehlswörter	940
Datentypen	942
Operatoren	943

Die folgende Tabelle zeigt einen repräsentativen Ausschnitt der Syntaxelemente der beiden Programmiersprachen im Vergleich. Diese Tabelle ist nicht vollständig.

## Befehlswörter

Zweck	Visual Basic 2010	C# 2010
Einsprungpunkt	Sub Main(ByVal args() As String)	static void Main(string[] args)
Namensraum	Namespace X ... End Namespace	namespace X { ... }
Klasse	Class ... End Class	class { ... }
Öffentliche Klasse	Public Class	public class
Klasse nur innerhalb der Assembly sichtbar	Friend Class	internal class
Partielle Klasse	Partial Class	partial class
Variablendeklaration/ Attributdefinition als Field	Dim x as Typ	Typ x
Attributdefinition als Property	Property X() As String Get Return _X End Get Set(ByVal value As String) _X = value End Set End Property	public string X { get { return x; } set { x = value; } } oder kurz: public Type Name { get; set; }
Attributdefinition als Automatic Property	Public Property x As String	public long x { get; set; }
Array	Dim x as Byte()	byte[] x;
Array-Größenveränderung	ReDim Preserve	Array.Resize()
Methode mit Rückgabewert	Function f() as Typ ... End Function	Typ f() { ... }
Methode ohne Rückgabewert	Sub f() as Typ ... End Sub	void f() { ... }
Überladene Methode	Overloads	(keine Zusatzangabe)
Methode verlassen	Return	return ►

Zweck	Visual Basic 2010	C# 2010
Methode verlassen und beim nächsten Aufruf danach fortsetzen	n/a	yield
Bezug auf Basisklasse	MyBase	base
Bezug auf aktuelle Klasse	MyClass	(Name der Klasse)
Bezug auf das aktuelle Objekt	Me	this
Implementierungsvererbung	Inherits	class C1 : C2
Deklaration einer Schnittstelle	Interface	interface
Schnittstellenvererbung	Implements	class C1 : I1
Konstantes Mitglied	Const	const
Methoden ohne Rückgabewert	Sub	void
Statisches Mitglied	Shared	static
Enumeration	Enum <members> End Enum	enum
Abstrakte Klasse	MustInherit	abstract
Finale Klasse	NotInheritable	sealed
Überschreiben einer Methode	Overrides	override
Abstrakte Methode	MustOverride	abstract
Versiegelte Methode	NotOverridable	sealed
Überschreibbare Methode	Overridable	virtual
Verdeckendes Mitglied	Shadows	(keine Zusatzangabe)
Konstruktor	Sub New() ... End Sub	public Klassenname() { ... }
Destruktor/Finalizer	Sub Finalize() ... End Sub	~Person() { ... }
Referenz auf eine Methode	Delegate	delegate
Mitglied mit Ereignissen	WithEvents	n/a
Bindung einer Ereignisbehandlungsroutine	Handles AddHandler RemoveHandler	+= -=
Blockbildung für Objekte	With obj ... End With	n/a
Datumsliteral	#12/24/2010#	New DateTime(2010,12,24)
Zeilenumbruch	vbCrLf	"\n"
Wertlose Werttypen	Nullable(Of Typ)	Typ? Oder Nullable<Typ>
Generische Klasse	Klasse(of Typ)	Klasse<Typ>
Typermittlung	obj.GetType()	typeof(obj) obj.GetType()
Typkonvertierung	CType() DirectCast TryCast	(Typ) Variable Variable as Type
Typvergleich	TypeOf k1 Is Kunde	k1 is Kunde ►

Zweck	Visual Basic 2010	C# 2010
Anonyme Methoden	n/a	<code>+= delegate() { ... }</code>
Zeigerprogrammierung	n/a	<code>unsafe, &amp;x, *x</code>
LINQ-Abfrageausdruck	<code>From m In Menge Where m.Feld &lt; 1000 Select m;</code>	<code>from m in Menge where m.Feld &lt; 1000 select m</code>
Implizit typisierte Variable	<code>Dim x = Wert</code>	<code>var x = Wert</code>
Lambda-Ausdruck	<code>Dim f As Func(Of String, Integer) = Function(s) s.Length</code>	<code>Func&lt;string, int&gt; f = s =&gt; s.Length;</code>
Lambda-Ausdruck, mehrzeilig, mit Rückgabewert	<code>Dim f = Function(x As Integer) Console.WriteLine(x) Return x &lt; 10 End Function</code>	<code>Predicate&lt;int&gt; f = x =&gt; { Console.WriteLine(x); return x &lt; 10; };</code>
Lambda-Ausdruck, mehrzeilig, ohne Rückgabewert	<code>Dim f = Sub(x) Trace.WriteLine(x) Console.WriteLine(x) End Sub</code>	<code>Action&lt;int&gt; f = x =&gt; { Trace.WriteLine(x); Console.WriteLine(x); };</code>
XML-Literale	<code>&lt;Element&gt;.&lt;Element&gt;.@Attribut</code>	n/a

## Datentypen

	Visual Basic 2010	C# 201
Ganzzahl 1 Byte	Byte	byte
Ganzzahl Boolean	Boolean	bool
Ganzzahl 2 Bytes	Short	short
Ganzzahl 4 Bytes	Integer	int
Ganzzahl 8 Bytes	Long	long
Zahl 4 Bytes	Single	float
Zahl 8 Bytes	Double	double
Zahl 12 Bytes	Decimal	decimal

	Visual Basic 2010	C# 201
Zeichen 1 Byte oder 2 Bytes	Char	char
Zeichenkette	String	string
Datum/Uhrzeit	Date	DateTime
Dynamischer Typ	Object	dynamic

# Operatoren

	Visual Basic 2010	C# 2010
<b>Mathematik</b>		
Addition	+	+
Subtraktion	–	–
Multiplikation	*	*
Division	/	/
Ganzzahldivision	\	/
Modulus	Mod	%
Potenz	^	n/a
Negation	Not	~
Inkrement	n/a	++
Dekrement	n/a	--
<b>Zuweisung</b>		
Einfache Zuweisung	=	=
Addition	+=	+=
Subtraktion	-=	-=
Multiplikation	*=	*=
Division	/=	/=
Ganzzahl-Division	\=	/=
Zeichenkettenverbindung	&=	+=
Modulus (Divisionsrest)	n/a	%=
Bit-Verschiebung nach links	<<=	<<=
Bit-Verschiebung nach rechts	>>=	>>=
Bit-weises UND	n/a	&=
Bit-weises XOR	n/a	^=
Bit-weises OR	n/a	=

	Visual Basic 2010	C# 2010
<b>Vergleich</b>		
Kleiner	<	<
Kleiner gleich	< =	< =
Größer	>	>
Größer gleich	> =	> =
Gleich	=	= =
Nicht gleich	< >	!=
Objektvergleich	Is	= =
Objektvergleich (negativ)	IsNot	!=
Objekttypvergleich	TypeOf x Is Class1	x is Class1
Zeichenkettenvergleich	=	= =
Zeichenkettenverbindung	&	+
<b>Logische Operatoren</b>		
UND	And	&&
ODER	Or	
NICHT	Not	!
Short-circuited UND	AndAlso	&&
Short-circuited ODER	OrElse	
<b>Bit-Operatoren</b>		
Bit-weises UND	And	&
Bit-weises XOR	Xor	^
Bit-weises OR	Or	
Bit-Verschiebung nach links	<<	<<
Bit-Verschiebung nach rechts	>>	>>
<b>Sonstiges</b>		
Bedingt	Iif-Funktion und If-Operator	?:
Bedingt (für Nullable Types)	n/a	?? :