

# Inhalt

<b>1</b>	<b>Strukturen (struct)</b>	<b>8</b>
1.1	Übersicht . . . . .	8
1.2	Fakten zu Strukturen . . . . .	8
1.2.1	Werttypen . . . . .	8
1.2.2	Default-Konstruktor . . . . .	8
1.2.3	Keine Vererbung . . . . .	8
1.2.4	Keine virtuellen Mitglieder . . . . .	8
1.2.5	Kopierverhalten . . . . .	9
1.2.6	Boxing und Unboxing . . . . .	9
1.2.7	Verwendung . . . . .	9
1.2.8	Performance . . . . .	9
1.3	Die Struktur DateTime . . . . .	10
1.4	Differenz zwischen zwei DateTime-Instanzen . . . . .	11
1.5	Übung Struct . . . . .	13
<b>2</b>	<b>Enum (Aufzählungstyp)</b>	<b>14</b>
2.1	Übersicht . . . . .	14
2.2	Definition . . . . .	14
2.3	Standardwerte . . . . .	14
2.4	Zugriff auf Enum-Werte . . . . .	14
2.5	Ganzzahlwerte . . . . .	15
2.6	Verwendung von Enums . . . . .	15
2.7	Enum-Methoden . . . . .	15
2.8	Flags-Attribute . . . . .	16
2.9	Beispiel . . . . .	17
2.10	Übung Enum . . . . .	18
2.10.1	Lösung . . . . .	19
<b>3</b>	<b>Interfaces in C#</b>	<b>20</b>
3.1	Übersicht . . . . .	20
3.2	Definition eines Interfaces . . . . .	20
3.3	Implementierung eines Interfaces . . . . .	21
3.4	Mehrfache Interface-Implementierung . . . . .	21
3.5	Explizite Interface-Implementierung . . . . .	22
3.6	Verwendung von Interfaces für Polymorphie . . . . .	22
3.6.1	Einschränkungen der Mehrfachvererbung . . . . .	22
3.7	Interface-Erweiterung . . . . .	23
<b>4</b>	<b>Generische Typen und Methoden</b>	<b>25</b>
4.1	Generische Typen in C# . . . . .	25
4.1.1	Vorteile von generischen Typen . . . . .	25
4.1.2	Syntax von generischen Typen: . . . . .	25

4.2	Generische Methoden in C#	27
4.2.1	Vorteile von generischen Methoden	27
4.2.2	Syntax von generischen Methoden	28
4.2.3	Einschränkungen von generischen Typen und Methoden mit 'where'	29
4.2.4	Beispiel für eine Einschränkung von generischen Typen	30
<b>5</b>	<b>Collections</b>	<b>31</b>
5.1	Übersicht	31
5.2	Häufig verwendete Collection Namespaces	31
5.2.1	System.Collections	31
5.2.2	System.Collections.Generic	31
5.2.3	System.Array	31
5.2.4	System.Linq	31
5.2.5	Weitere Collections Namespaces	32
5.3	Interfaces IEnumerable<T> und IEnumerator<T>	32
5.3.1	IEnumerable<T>	32
5.3.2	IEnumerator<T>	32
5.3.3	Verwendung von IEnumerable<T> und IEnumerator<T>	33
5.4	Funktionsweise von foreach	35
5.5	Die ICollection<T> und IList<T> Interfaces in C#	37
5.5.1	ICollection<T>	37
5.5.2	IList<T>	37
5.5.3	Verwendung	37
5.6	Die Array-Klasse	37
5.6.1	Erstellen von Arrays in C#	38
5.6.2	Verwendung der Array-Klasse	40
5.7	Die List<T>-Klasse	41
5.7.1	Eigenschaften und Methoden	41
5.7.2	Verwendung	41
5.8	Übung List<T>	43
5.8.1	Übung	43
5.8.2	Lösung	43
5.9	Die Queue<T>-Klasse	44
5.9.1	Eigenschaften und Methoden	44
5.9.2	Verwendung	44
5.10	Die Stack<T>-Klasse	46
5.10.1	Eigenschaften und Methoden	46
5.10.2	Verwendung	46
5.11	Das IDictionary<TKey, TValue>-Interface	47
5.11.1	Eigenschaften und Methoden	47
5.11.2	Verwendung	48
5.12	Die Dictionary<TKey, TValue>-Klasse	48
5.12.1	Eigenschaften und Methoden	48
5.12.2	Verwendung	49
5.13	Übung Dictionary	51
5.13.1	Übung	51
5.13.2	Lösung	52
5.14	Indexer	52
5.15	Zusammenfassung	53

<b>6</b>	<b>Modernes C#</b>	<b>54</b>
6.1	Das Schlüsselwort <code>var</code> , Objektinitialisierer und anonyme Typen in C#	54
6.1.1	Beispiel <code>var</code>	54
6.1.2	Beispiel Objektinitialisierer	54
6.1.3	Beispiel anonymer Typ	55
6.2	Initialisierer für Collections in C#	55
6.2.1	Beispiel für Initialisierer für Collections	55
6.3	Nullable Value Types	56
6.3.1	Operatoren für nullable Value Types	57
6.4	Nonnullable Reference Types	58
6.4.1	Relevante Operatoren	59
6.5	Extension Methods in C#	60
6.5.1	Eigenschaften von Extension Methods	60
6.5.2	Syntax von Extension Methods	60
6.5.3	Beispiel für eine Extension Methode	62
6.6	Übung Extension Methods	62
6.6.1	Übung	62
6.6.2	Lösung	63
6.7	Evolution der Codeeinspeisung in C#	63
6.7.1	Benannte Methoden	64
6.7.2	Anonyme Methoden	65
6.7.3	Lambda-Ausdrücke	66
6.7.4	Verwendung	67
6.8	Expression Bodies	68
6.8.1	Methode mit Blocknotation	68
6.8.2	Methode mit Expression Body	69
6.8.3	Property mit Blocknotation	70
6.8.4	Property mit Expression Body	71
6.9	Tupel in C#	71
6.9.1	Tupel-Erstellung:	71
6.9.2	Zugriff auf Tupel-Elemente:	72
6.9.3	Benannte Tupel-Elemente:	72
6.10	Der Null-conditional Operator	73
6.10.1	Verwendung des Null-conditional Operators:	73
6.10.2	Vollständiges Beispiel	74
6.10.3	Kaskadierung des Null-conditional Operators:	74
6.11	Range-Operator in C# 8	75
6.11.1	Verwendung des Range-Operators	75
6.11.2	Eigenschaften des Range-Operators	75
6.12	Nameof Expressions	76
6.12.1	Verwendung von nameof expressions	76
6.12.2	Verwendungszwecke von nameof expressions	76
6.12.3	Reales Beispiel für die Verwendung von nameof Expressions	77
6.13	Lokale Funktionen	77
6.13.1	Eigenschaften von Lokalen Funktionen	78
6.13.2	Beispiel für Lokale Funktionen	78
6.14	Übung Lokale Funktionen	78
6.14.1	Übung	78
6.14.2	Lösung	80

6.15	'yield'-return	80
6.15.1	Hauptfunktion von 'yield'	80
6.15.2	Beispiel für 'yield'	81
6.16	Read-only Auto-Property mit automatischer Initialisierung in C# 9.0	82
6.16.1	Codebeispiel und Erklärung	82
6.17	Records und init	83
6.17.1	Vergleichen von Objekten und Strings in C#	83
6.17.2	Records und init	85
6.17.3	Die automatische Implementierung von Gleichheitsvergleichen	85
<b>7</b>	<b>Strings</b>	<b>87</b>
7.1	Einführung in C# Strings	87
7.1.1	Die String-Klasse	87
7.1.2	Verwendung von Interfaces mit der <code>string</code> -Klasse	87
7.1.3	<code>sealed</code>	88
7.1.4	Methoden und Eigenschaften	89
7.1.5	String-Erstellung	89
7.1.6	String-Operatoren und -Methoden	89
7.1.7	Unveränderlichkeit von Strings	89
7.2	Wichtige Methoden der <code>String</code> -Klasse	90
7.2.1	Verkettung und Formatierung	90
7.2.2	Suchen und Vergleichen	91
7.2.3	Extrahieren und Teilen	91
7.2.4	Ändern und Entfernen	92
7.2.5	Prüfen und Konvertieren	93
7.3	Reguläre Ausdrücke	96
7.3.1	Einführung in reguläre Ausdrücke (Regex)	96
7.3.2	Musterabgleich (Match)	97
7.3.3	Alle Vorkommen eines Musters finden (Matches)	98
7.3.4	Ersetzen von Mustern	98
7.3.5	Aufteilen anhand von Mustern	99
7.3.6	Prüfen auf Übereinstimmung	99
7.4	Übung Regex	99
7.4.1	Übung	99
7.4.2	Lösung	100
<b>8</b>	<b>Exceptions</b>	<b>101</b>
8.1	Übersicht über Exceptions in C#	101
8.1.1	Arten von Exceptions	101
8.2	Behandlung von Exceptions	102
8.3	Eine Exception produzieren	102
8.4	Vorteile von Exception-basierter Fehlerbehandlung in C#	104
8.4.1	Trennung von Fehlerbehandlungscode und Hauptcode	104
8.4.2	Bessere Fehlerlokalisierung und Debugging	104
8.4.3	Mehr Flexibilität in der Fehlerbehandlung	104
8.4.4	Stack Unwinding	104
8.5	Werfen von vordefinierten Exceptions	104
8.6	Spezifische Exception Handler	105
8.7	Benutzerdefinierte Exceptions	107

---

<b>9</b>	<b>Pattern Matching</b>	<b>109</b>
9.1	Übersicht	109
9.1.1	C# 7.0 Pattern Matching mit is und switch	109
9.1.2	C# 7.0 Erweitertes Pattern Matching mit switch	109
9.1.3	C# 8.0 Switch Expressions	109
9.1.4	C# 9.0 Relational and Logical Patterns	109
9.1.5	C# 9.0 Type Patterns with and, or, not	109
9.1.6	C# 10.0 Match Expressions	109
9.1.7	Zusammenfassung	110
9.2	C# 7.0 Pattern Matching mit is und switch	110
9.2.1	C# 7.0 Pattern Matching mit is	110
9.2.2	C# 7.0 Pattern Matching mit switch	111
9.2.3	C# 7.0 Erweitertes Pattern Matching mit switch	112
9.2.4	C# 8.0 Switch Expressions	113
9.2.5	C# 9.0 Relational and Logical Patterns	114
9.2.6	C# 9.0 Type Patterns with and, or, not	115
9.2.7	C# 10.0 Match Expressions	116
<b>10</b>	<b>Delegates und Events</b>	<b>117</b>
10.1	Einführung in Delegates	117
10.2	Delegate-Instanzierung und Verwendung	117
10.3	Multicast-Delegates	119
10.4	Events	120
10.5	Delegates und Events in einem gemeinsamen Beispiel	122
10.6	Best Practices	123
10.6.1	Best Practices	123
10.7	Vordefiniertes EventHandler-Delegate in C#	123
10.8	Die Func und Action Delegates in C#	124
10.9	Beispiel nach dem Umbau für Best Practices und EventHandler delegate (relevante Teile)	126
10.10	Übung Events	127
10.10.1	Übung	127
10.10.2	Lösung	127
<b>11</b>	<b>Asynchrone Programmierung</b>	<b>129</b>
11.1	Einführung in asynchrone Programmierung	129
11.1.1	Erklärung von asynchroner Programmierung und deren Vorteile	129
11.1.2	Warum und wann sollte asynchrone Programmierung verwendet werden?	129
11.2	async und await Schlüsselwörter	129
11.2.1	Erklärung von async und await	129
11.2.2	Wie werden sie verwendet? Grundlegende Syntax und Verwendung	129
11.3	Task-basierte asynchrone Programmierung	130
11.3.1	Verwendung von Task und Task<T> zur Erstellung asynchroner Methoden	130
11.3.2	Rückgabe von Ergebnissen aus asynchronen Methoden	131
11.4	Parallelität und asynchrone Programmierung	131
11.4.1	Vergleich von paralleler Programmierung und asynchroner Programmierung	131
11.4.2	Verwendung von Parallelität in asynchronen Szenarien	131
<b>12</b>	<b>Service-Oriented Architecture und APIs</b>	<b>133</b>
12.1	Übersicht	133

---

12.2	Application Programming Interfaces (APIs)	133
12.3	RESTful APIs: Merkmale und Kriterien	134
12.4	Ein praktisches Beispiel	135
12.4.1	Ein API im Browser testen	135
12.4.2	Das API aus einem C#-Programm aufrufen	137
12.4.3	Die Antwort des Servers verarbeiten	140
12.4.4	Datenobjekte verbessern	142
12.5	WSDL und OpenAPI	143
12.5.1	WSDL (Web Services Description Language)	143
12.5.2	OpenAPI (ehemals Swagger)	143
12.5.3	Unterschiede und Verwendung	144
12.5.4	Verwendungsmöglichkeiten einer OpenAPI-Spezifikation	144
<b>13</b>	<b>LINQ</b>	<b>145</b>
13.1	Übersicht	145
13.2	Übersicht über Language Integrated Query (LINQ)	145
13.2.1	Typen von LINQ	145
13.2.2	Grundlegende Konzepte von LINQ	145
13.2.3	Beispiel einer LINQ-Abfrage	146
13.3	Kategorien von LINQ-Abfrageoperatoren	147
13.3.1	Filtering Operators (Filteroperatoren)	147
13.3.2	Sorting Operators (Sortieroperatoren)	147
13.3.3	Projection Operators (Projektionsoperatoren)	147
13.3.4	Set Operators (Set-Operatoren)	148
13.3.5	Partitioning Operators (Partitionierungsoperatoren)	148
13.3.6	Element Operators (Elementoperatoren)	148
13.4	Infrastruktur für die folgenden Beispiele	148
13.4.1	Klasse Product	149
13.4.2	Klasse Infrastructure	150
13.4.3	Ausgabe der Produkte	151
13.5	Filteroperatoren	151
13.6	Sortieroperatoren	153
13.7	Projektionsoperatoren	155
13.8	Set-Operatoren	156
13.9	Elementoperatoren	157
13.10	Partitionierungsoperatoren	159
<b>14</b>	<b>WPF Tutorial</b>	<b>163</b>
14.1	Übersicht	163
14.2	XAML Code	163
14.3	Das Code-Behind File MainWindow.xaml.cs	165
14.4	Übung: Ellipsen	167
14.5	Lösung	169