6.3.2 ArrayList

6.3.3 Hashtable

6.3.4 Indexer

## 6.4 Generics

6.4.1 Generics bieten Typsicherheit

6.4.2 Generische Methoden

6.4.3 yield – Iteratoren

## 6.5 Generische Collections

6.5.1 List-Collection statt ArrayList

6.5.2 Vorteile generischer Collections

6.5.3 Constraints

## 6.6 Das Prinzip der Delegates

6.6.1 Delegates sind Methodenzeiger

6.6.2 Einen Delegate-Typ deklarieren

6.6.3 Ein Delegate-Objekt erzeugen

6.6.4 Anonyme Methoden

6.6.5 Lambda-Ausdrücke

6.6.6 Lambda-Ausdrücke in der Task Parallel Library

6.6.7 Action<> und Func<>

## 6.7 Dynamische Programmierung

6.7.1 Wozu dynamische Programmierung?

6.7.2 Das Prinzip der dynamischen Programmierung

6.7.3 Optionale Parameter sind hilfreich

6.7.4 Kovarianz und Kontravarianz

### 6.8 Weitere Datentypen

6.8.1 BigInteger

6.8.2 Complex

6.8.3 Tuple<>

6.8.4 SortedSet<>

### 6.9 Praxisbeispiele

6.9.1 ArrayList versus generische List

6.9.2 Generische IEnumerable-Interfaces implementieren

6.9.3 Delegates, Func, anonyme Methoden, Lambda Expressions

## 7 Einführung in LINQ

### 7.1 LINQ-Grundlagen

7.1.1 Die LINQ-Architektur

7.1.2 Anonyme Typen

7.1.3 Erweiterungsmethoden

### 7.2 Abfragen mit LINQ to Objects

7.2.1 Grundlegendes zur LINQ-Syntax

7.2.2 Zwei alternative Schreibweisen von LINQ-Abfragen

7.2.3 Übersicht der wichtigsten Abfrageoperatoren

## 7.3 LINQ-Abfragen im Detail

7.3.1 Die Projektionsoperatoren Select und SelectMany

7.3.2 Der Restriktionsoperator Where

7.3.3 Die Sortierungsoperatoren OrderBy und ThenBy

7.3.4 Der Gruppierungsoperator GroupBy

7.3.5 Verknüpfen mit Join

7.3.6 Aggregat-Operatoren

7.3.7 Verzögertes Ausführen von LINQ-Abfragen

7.3.8 Konvertierungsmethoden

7.3.9 Abfragen mit PLINQ

## 7.4 Praxisbeispiele

7.4.1 Die Syntax von LINQ-Abfragen verstehen

7.4.2 Aggregat-Abfragen mit LINQ

7.4.3 LINQ im Schnelldurchgang erlernen

7.4.4 Strings mit LINQ abfragen und filtern

7.4.5 Duplikate aus einer Liste entfernen

7.4.6 Arrays mit LINQ initialisieren

7.4.7 Arrays per LINQ mit Zufallszahlen füllen

7.4.8 Einen String mit Wiederholmuster erzeugen

7.4.9 Mit LINQ Zahlen und Strings sortieren

7.4.10 Mit LINQ Collections von Objekten sortieren

7.4.11 Where – Deep Dive

# 8 Neuerungen von C# im Überblick

## 8.1 C# 4.0 – Visual Studio 2010

8.1.1 Datentyp dynamic

8.1.2 Benannte und optionale Parameter

8.1.3 Kovarianz und Kontravarianz

## 8.2 C# 5.0 – Visual Studio 2012

8.2.1 Async und Await

8.2.2 CallerInfo

## 8.3 Visual Studio 2013

## 8.4 C# 6.0 – Visual Studio 2015

8.4.1 String Interpolation

8.4.2 Schreibgeschützte AutoProperties

8.4.3 Initialisierer für AutoProperties

8.4.4 Expression Body Member

8.4.5 using static

8.4.6 Bedingter Nulloperator

8.4.7 Ausnahmenfilter

8.4.8 nameof-Ausdrücke

8.4.9 await in catch und finally

8.4.10 Indexinitialisierer

## 8.5 C# 7.0 – Visual Studio 2017

8.5.1 out-Variablen

8.5.2 Tupel

8.5.3 Mustervergleich

8.5.4 Discards

8.5.5 Lokale ref-Variablen und Rückgabetypen

8.5.6 Lokale Funktionen

8.5.7 Mehr Expression Body Member

8.5.8 throw-Ausdrücke

8.5.9 Verbesserung der numerischen literalen Syntax

## 8.6 C# 7.1 bis 7.3 – Visual Studio 2019

8.6.1 C# 7.1

8.6.2 C# 7.2

8.6.3 C# 7.3

8.6.4 Visual Studio 2019 – Live Share

## 8.7 C# 8.0

8.7.1 Standardschnittstellenmethoden

8.7.2 Vereinfachung von switch-Ausdrücken

8.7.3 Eigenschaftenmuster