C# Programmieren für Einsteiger

Michael Bonacina

2. Auflage: Februar 2019

ISBN 978-3-96645-001-0

Herausgegeben durch:
BMU Media GmbH
Hornissenweg 4
84034 Landshut

C# Programmieren für Einsteiger

Inhaltsverzeichnis

Alle Programmcodes aus diesem Buch sind als PDF zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:

http://bmu-verlag.de/cs-programmieren/

Kapitel 1
Einleitung

Eine Programmiersprache zu erlernen, mag eine große Herausforderung darstellen. Wenn man sich jedoch darauf einlässt, dann stellt man in der Regel fest, dass die ersten Schritte nicht allzu schwer sind. Wer sich dieser Aufgabe stellt, profitiert von enormen Vorteilen. Programmierkenntnisse sind beispielsweise bei der Suche nach einer Arbeitsstelle sehr hilfreich. Die Digitalisierung schreitet stetig voran und immer mehr Arbeitgeber benötigen Fachkräfte, die in der Lage dazu sind, Computerprogramme zu erstellen. Doch selbst Arbeitnehmer, die nicht direkt als Programmierer tätig sind, profitieren stark von diesen Kenntnissen. Wer gut programmieren kann, versteht die Prozesse und Abläufe der Digitalisierung deutlich besser und kann dadurch besser zum Wachstum des Unternehmens beitragen. Die Mühe, die es mit sich bringt, eine Programmiersprache zu erlernen, macht sich daher durch eine deutliche Verbesserung der Karrierechancen schnell bezahlt. Um das Programmieren zu erlernen, ist es zunächst notwendig, eine Programmiersprache auszuwählen. Dafür gibt es viele verschiedene Möglichkeiten. Dieses Buch befasst sich mit der Programmiersprache C# (was als C Sharp ausgesprochen wird). Diese stellt eine ausgezeichnete Wahl dar, um das Programmieren von Grund auf zu erlernen. Die folgenden Abschnitte stellen vor, was C# auszeichnet und welche Unterschiede zu anderen Programmiersprachen bestehen.

1.1 C#: Eine Programmiersprache mit vielfältigen Anwendu ngsmöglichkeiten

Einer der großen Vorteile von C# besteht darin, dass sich diese Programmiersprache für sehr viele verschiedene Anwendungen eignet. Hierbei handelt es sich um eine General Purpose Language – auf Deutsch um eine Allzweck- oder eine Mehrzweck Programmiersprache. Das bedeutet, dass sich damit ganz unterschiedliche Probleme lösen lassen – von Anwenderprogrammen und Web-Applikationen bis hin zur Hardwareprogrammierung.

Darüber hinaus ist es möglich, mit C# Programme für verschiedene Plattformen zu gestalten. Da diese Programmiersprache von Microsoft entwickelt wurde, kommt sie zwar vorwiegend in einer Windows-Umgebung

zum Einsatz. Mittlerweile bestehen jedoch auch zahlreiche Möglichkeiten, um mit C# Programme für Computer zu schreiben, die die Betriebssysteme macOS oder Linux verwenden. Selbst für mobile Endgeräte, die mit den Betriebssystemen Android oder iOS arbeiten, lassen sich mit C# Programme gestalten. Auch dieser Aspekt trägt dazu bei, dass C# sich universell einsetzen lässt.

Schließlich unterstützt C# viele verschiedene Programmierparadigmen. Es ist damit möglich, sowohl imperative als auch deklarative Programme zu erstellen. C# bietet eine volle Unterstützung für die objektorientierte Programmierung und lässt auch eine funktionale Programmierung zu. Anfänger werden sicherlich nicht wissen, was hinter all diesen Ausdrücken steckt. Das ist im Moment auch noch nicht von Bedeutung. Wichtig ist es lediglich, zu wissen, dass es sich bei C# um eine Multiparadigmensprache handelt. Das erlaubt es dem Programmierer später, sich auf eine bestimmte Programmiertechnik zu spezialisieren oder das verwendete Paradigma individuell an die Aufgaben anzupassen.

Zusammenfassend lässt sich sagen, dass sich C# beinahe für jede Aufgabe im Bereich der Programmierung eignet. Das ist für den Anfänger sehr vorteilhaft. Auf diese Weise kann er zunächst die grundlegenden Programmiertechniken erlernen. Später ist es möglich, sich auf jeden beliebigen Bereich zu spezialisieren, ohne die Programmiersprache zu wechseln.

1.2 Die Entwicklungsgeschichte von C#

Die Entwicklung von C# hat ihren Ursprung in den letzten Jahren des vergangenen Jahrhunderts. Die Sprache steht in engem Zusammenhang mit der Entwicklung des .NET-Frameworks. Dabei handelt es sich um eine Plattform, die unter anderem aus einer Laufzeitumgebung für die Ausführung von Anwenderprogrammen besteht. Darüber hinaus sind viele weitere Bestandteile enthalten, die es erlauben, Programme für diese Umgebung zu entwickeln.

Microsoft verwendeten zuvor andere Techniken für diese Aufgaben. Diese waren jedoch bereits in die Jahre gekommen und entsprachen nicht mehr dem Stand der Technik. Daher entschloss sich der Softwarekonzern aus den USA dazu, ein modernes System zu entwerfen. Das führte dazu, dass er mit der Entwicklung des .NET-Frameworks begann.

Dabei kam es zu einem weiteren Problem. Microsoft verwendete bis zu diesem Zeitpunkt vorwiegend die Programmiersprache Java für die Ent-

wicklung von Anwendungsprogrammen. Dieses System stammte jedoch ursprünglich vom Unternehmen Sun Microsystems. Microsoft adaptierte diese Programmiersprache und erweiterte sie anhand der eigenen Bedürfnisse. Damit waren die Entwickler bei Sun jedoch nicht einverstanden und strebten daher ein Gerichtsverfahren an. Das führte dazu, dass sich Microsoft dazu entschloss, für das .NET-Framework eine neue Programmiersprache zu entwickeln.

Die Arbeit an diesem Projekt begann im Januar 1999 unter Leitung des bekannten Informatikers Anders Hejlsberg. Es erhielt zunächst den Namen Cool – als Abkürzung für C-like Object Oriented Language. Daran wird bereits deutlich, an welchen Grundsätzen sich C# orientierte. Zum einen sollte eine Programmiersprache entstehen, die auf der Syntax von C basiert. C war zu diesem Zeitpunkt noch eine der am häufigsten verwendeten Programmiersprachen. Darüber hinaus sollte die neue Sprache objektorientiert sein. Dabei handelt es sich um eines der wichtigsten Konzepte der modernen Informatik. Die Grundlagen der objektorientierten Programmierung werden auch später in diesem Buch noch behandelt. Auf diese Weise entstand eine neue Programmiersprache, deren Aufbau vielen Programmierern der damaligen Zeit aufgrund der Ähnlichkeit zu C bereits bekannt vorkam. Dennoch setzte sie alle Anforderungen an eine moderne Sprache um.

Die Vorstellung der Programmiersprache fand 2001 statt. Microsoft verabschiedete sich jedoch kurz vor der Präsentation von der Bezeichnung Cool, da bereits andere Unternehmen diese Markenbezeichnung nutzten. Daher wählte es die Bezeichnung C#. Das Rautensymbol leitet sich vom Kreuzzeichen aus der Notenschrift ab. Dieses steht für die Erhöhung um einen Halbton. Das soll symbolisieren, dass es sich bei C# um eine geringfügige Weiterentwicklung von C handelt. Die erste Version wies jedoch sehr starke Ähnlichkeiten zur Programmiersprache Java auf. Doch auch Einflüsse anderer Sprachen waren bemerkbar – beispielsweise von C++, Haskell oder Delphi (das ebenfalls von Anders Hejlsberg entwickelt wurde). Die starke Ähnlichkeit zu Java wurde erst 2005 mit der Vorstellung von C# 2.0 aufgehoben.

1.3 C# und .NET Framework

Wie bereits aus dem vorherigen Abschnitt hervorgeht, besteht ein enger Zusammenhang zwischen C# und dem .NET Framework. Daher ist es sinnvoll,

kurz vorzustellen, was das .NET Framework überhaupt ist und welcher Art die Verbindung zur Programmiersprache C# ist.

Das .NET Framework entstand zu Beginn des neuen Jahrtausends. Die erste Version erschien 2002. Dabei handelt es sich um ein Framework, das es ermöglicht, Anwenderprogramme zu entwickeln und auszuführen. Da der Softwarekonzern Microsoft für dessen Entwicklung verantwortlich war, liegt es auf der Hand, dass diese Software für die Verwendung unter dem Betriebssystem Windows vorgesehen ist.

Das .NET Framework besteht zum einen aus einer Laufzeitumgebung. Dabei handelt es sich um ein Programm, das die Ausführung eines Anwenderprogramms ermöglicht. Um ein Programm auf einem Computer auszuführen, ist es notwendig, die einzelnen Kommandos, die in Textform vorliegen, in die Maschinensprache zu übersetzen. Diese Übersetzung wird als Kompilierung bezeichnet. Bei der Ausführung eines Programms in C# handelt es sich jedoch um einen zweistufigen Prozess. Zunächst kommt es zur Übersetzung in eine Zwischensprache. Diese weist zwar bereits große Ähnlichkeiten zur Maschinensprache auf, doch ist sie noch unabhängig von der Architektur des Rechners und vom verwendeten Betriebssystem. Erst bei der Ausführung wird diese Zwischensprache in die eigentliche Maschinensprache übersetzt. Diese Aufgabe übernimmt die Laufzeitumgebung.

Darüber hinaus enthält das .NET Framework Programmierschnittstellen, Klassenbibliotheken und anderen Werkzeugen, die das Programmieren einfacher gestalten. Außerdem ist eine Entwicklungsumgebung im .NET Framework inbegriffen. Dabei handelt es sich um ein Programm, das alle notwendigen Werkzeuge zum Programmieren bereitstellt. Es besteht aus einer Oberfläche, in die der Programmierer seinen Code direkt einfügen kann. Des Weiteren sind jedoch noch viele weitere Werkzeuge enthalten – beispielsweise um Fehler im Programmcode zu beseitigen oder um fertige Codebausteine anhand einer grafischen Benutzeroberfläche einzufügen. Der Name dieser Entwicklungsumgebung lautet Visual Studio.

Das .NET Framework ist ausschließlich für das Betriebssystem Windows erhältlich. Da dieses für die Entwicklung und die Ausführung von C#-Programmen ursprünglich notwendig war, kam diese Programmiersprache zu Beginn fast ausschließlich für die Programmierung unter Windows zum Einsatz. Mittlerweile gibt es jedoch auch Alternativen für andere Betriebssysteme. Daher ist es inzwischen möglich, mit C# Programme für Linux, macOS und für weitere Plattformen zu erstellen.

1.4 Vom Code zum fertigen Programm: die Kompilierung

Bereits im vorherigen Abschnitt wurde kurz angesprochen, dass es für die Ausführung eines Programms notwendig ist, dieses in die Maschinensprache zu übersetzen. Dieser Prozess ist für die Erstellung des Programms von enormer Bedeutung. Daher soll er an dieser Stelle etwas ausführlicher vorgestellt werden.

Wenn ein Computer ein Programm ausführt, sendet er ein Kommando aus einzelnen Bits an den Prozessor. Jeder dieser Befehle hat eine bestimmte Ausgabe des Prozessors zur Folge. Sie werden als Maschinensprache bezeichnet. Ein ausführbares Programm besteht aus einer festen Abfolge dieser Befehle in Maschinensprache. Wenn man jedoch eine gewöhnliche Programmiersprache betrachtet, ist diese ganz anders aufgebaut. Sie besteht aus Zahlen und aus bestimmten Schlüsselbegriffen. Bevor es möglich ist, das Programm auszuführen, ist es notwendig, es in die Maschinensprache zu übersetzen.

Dafür gibt es grundsätzlich zwei Alternativen. Zum einen ist es möglich, diese Übersetzung ein einziges Mal vorzunehmen und auf diese Weise ein ausführbares Programm zu erzeugen. Dieser Vorgang wird als Kompilierung bezeichnet. Eine andere Vorgehensweise besteht darin, die Übersetzung jedes Mal aufs Neue vorzunehmen. Diese Methode wird als Interpretierung bezeichnet. Dabei entsteht kein ausführbares Programm. Der Code lässt sich nur mithilfe eines speziellen Übersetzungsprogramms – der als Interpreter bezeichnet wird – ausführen.

Beide Möglichkeiten bieten einige Vor- und Nachteile. Kompilierte Programme arbeiten beispielsweise sehr effizient. Wenn der Code bereits in Maschinensprache vorliegt, dann begünstigt das eine schnelle Ausführung. Allerdings lassen sich die Programme nicht auf andere Rechnerarchitekturen oder Betriebssysteme übertragen. Interpretierte Programme arbeiten etwas ineffizienter, da bei jeder Ausführung eine erneute Übersetzung notwendig ist. Allerdings sind sie deutlich unabhängiger. Sie lassen sich auf jedem Computer ausführen, der über einen entsprechenden Interpreter verfügt.

C# verwendet hierbei eine Zwischenlösung. Zunächst ist es notwendig, das Programm zu kompilieren. Dabei entsteht jedoch noch keine Maschinensprache, sondern eine Zwischensprache. Für die Ausführung ist dann die .NET Laufzeitumgebung notwendig. Dabei handelt es sich um einen Interpreter, der das Programm während der Laufzeit in die Maschinensprache überträgt. Da bei der Erstellung der Zwischensprache jedoch bereits zahlreiche Arbeitsschritte abgeschlossen wurden, verläuft dieser Prozess deut-

lich schneller als bei gewöhnlichen interpretierten Sprachen. Das führt zu einer effizienten Ausführung und ermöglicht es dennoch, die Programme auf unterschiedlichen Architekturen auszuführen – vorausgesetzt sie verfügen über die .NET Laufzeitumgebung. Auf diese Weise verbindet C# die Vorteile beider Methoden.

Kapitel 2

Visual Studio: die passende IDE für die Entwicklung mit C#

Um ein Computerprogramm zu erzeugen und auszuführen, sind drei Schritte notwendig:

1. das Programm schreiben
2. das Programm kompilieren (in die C#-Zwischensprache vorübersetzen)
3. das Programm auszuführen

Für jeden dieser Schritte sind passende Hilfsmittel erforderlich.

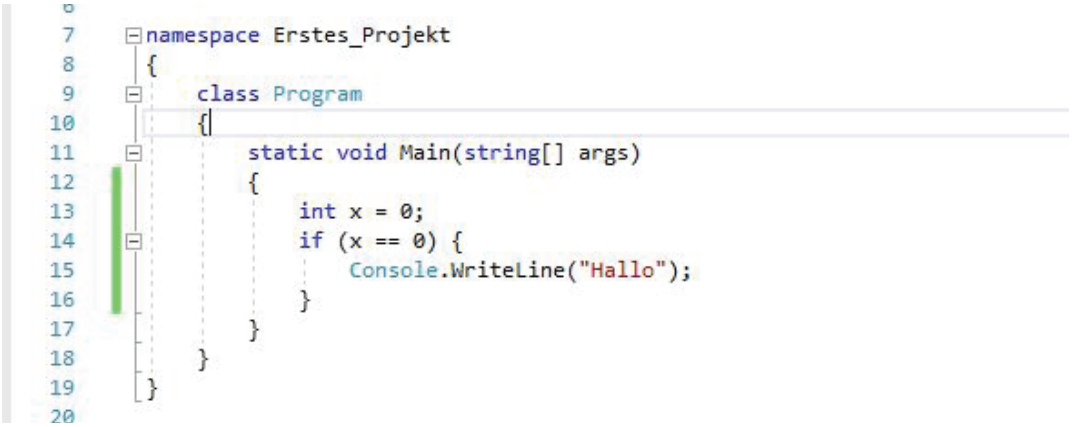
Für die erste Aufgabe kommt ein Texteditor zum Einsatz. Dabei handelt es sich um ein Programm, das die Eingabe von Text ermöglicht. Im Gegensatz zu gewöhnlichen Textverarbeitungsprogrammen wie Word speichern Texteditoren die Inhalte jedoch als reinen Text ab – ohne weitere Formatierungen. Das ist erforderlich, damit der Compiler die Inhalte später versteht. Darüber hinaus ermöglicht es der Texteditor, ein passendes Dateiformat für C# zu wählen. Wenn man den Code mit einem gewöhnlichen Textverarbeitungsprogramm erstellt, ist es später nicht möglich, das Programm auszuführen. Daher ist es notwendig, über einen geeigneten Texteditor zu verfügen. Die meisten Betriebssysteme enthalten bereits eine entsprechendeSoftware. Unter Windows steht beispielsweise der Microsoft Editor – auch als Notepad bekannt – zur Verfügung. Mit diesem ist das Programmieren zwar grundsätzlich möglich, doch bietet er nur einen minimalen Funktionsumfang. Daher ist es ratsam, einen etwas leistungsfähigeren Texteditor zu verwenden.

Der zweite Schritt besteht in der Kompilierung. Diese ist erforderlich, um aus dem Quellcode ein lauffähiges Programm zu erzeugen. Auch hierfür ist eine passende Software notwendig – die als Compiler bezeichnet wird. Dieser übersetzt den Programmcode in die C#-Zwischensprache. Daher ist es notwendig, einen Compiler zu installieren, bevor man mit dem Programmieren beginnen kann.

Für die Ausführung des Programms ist schließlich die .NET-Laufzeitumgebung erforderlich. Dieser Punkte spielt für die meisten Leser wahrscheinlich keine große Rolle: Die .NET-Laufzeitumgebung ist bereits standardmäßig auf allen neueren Windows-Rechnern installiert. Lediglich Anwender, die ein anderes Betriebssystem verwenden, sollten überprüfen, ob eine entsprechende Software vorhanden ist.

Es ist möglich, den C#-Compiler und den Texteditor einzeln zu installieren. Allerdings gibt es auch eine praktische Möglichkeit, die bereits beide Bestandteile enthält: Visual Studio von Microsoft. Hierbei handelt es sich um eine integrierte Entwicklungsumgebung (Integrated Development Environment – IDE). Das bietet nicht nur den Vorteil, dass man dabei nur ein einziges Programm installieren muss. Darüber hinaus enthält die IDE noch viele weitere nützliche Funktionen für die Programmierung. Visual Studio und C# stammen beide von Microsoft und sind daher perfekt aufeinander abgestimmt. Daher eignet sich die Software hervorragend für die Arbeit mit dieser Programmiersprache. Ein weiterer Vorteil besteht darin, dass sie in der Community-Edition kostenlos zur Verfügung steht. Die folgenden Absätze stellen einige nützliche Funktionen von Visual Studio vor.

Die IDE erzeugt beim Eingeben des Programmcodes eine automatische Syntaxhervorhebung. Das bedeutet, dass bestimmte Befehle farblich im Quellcode gekennzeichnet werden. Das sorgt für eine deutlich übersichtlichere Gestaltungsweise und macht es einfacher, den erzeugten Code zu verstehen und darin nach Fehlern zu suchen. Außerdem ist es möglich, einzelne Programmteile, die einen in sich geschlossenen Block darstellen, einzuklappen. Das ist insbesondere bei längeren Programmen sinnvoll, um die Übersichtlichkeit zu verbessern. Bei Bedarf ist es jedoch jederzeit möglich, die entsprechenden Stellen wieder auszuklappen, um die Details zu betrachten. Die Software erzeugt außerdem automatische Einrückungen, um zusammengehörige Programmteile zu kennzeichnen. Auch das trägt zu einer übersichtlichen Gestaltungsweise bei. Schließlich bietet die IDE die Möglichkeit zur Autovervollständigung der Befehle. Wenn man die ersten Buchstaben eines Kommandos eingibt, präsentiert die Software mehrere passende Möglichkeiten. Diese kann man dann mit der Maus oder mit den Pfeiltasten auswählen und in das Programm einfügen. Das ermöglicht eine sehr effiziente Arbeitsweise.



```
7   namespace Erstes_Projekt
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              int x = 0;
14              if (x == 0) {
15                  Console.WriteLine("Hallo");
16              }
17          }
18      }
19  }
20
```

Screenshot 1 Der Programmcode mit Syntaxhervorhebung

Die bisher beschriebenen Vorteile beziehen sich nicht ausschließlich auf integrierte Entwicklungsumgebungen wie Visual Studio. Auch die meisten Texteditoren bieten die entsprechenden Möglichkeiten. Allerdings gibt es dabei auch Ausnahmen wie den Microsoft Editor, der diese Funktionen nicht enthält. Eine IDE wie Visual Studio bietet jedoch noch einige weitere Vorteile. Dazu zählt, dass diese Software die Funktionen eines Texteditors und des Compilers miteinander vereint. Wenn man diese Programme einzeln installiert, ist es für jeden Kompilierungsprozess notwendig, das Fenster zu wechseln. Da es während der Testphase notwendig ist, das Programm sehr oft zu kompilieren, um zu überprüfen, ob es alle Anforderungen erfüllt, geht dabei viel Zeit verloren. Visual Studio präsentiert jedoch beide Funktionen im gleichen Fenster. Obwohl die Zeitersparnis bei jeder einzelnen Kompilierung nur wenige Sekunden beträgt, erhöht sich in der Summe die Effizienz beim Programmieren dadurch erheblich. Darüber hinaus ist es bei einem herkömmlichen Compiler notwendig, den Befehl für die Kompilierung von Hand einzutippen. Auch das nimmt einige zusätzliche Sekunden in Anspruch. Bei Visual Studio reicht hierfür hingegen ein einziger Mausklick aus.

Darüber hinaus bietet die IDE noch viele weitere Zusatzfunktionen. Diese sind zwar für Anfänger meistens von geringer Bedeutung. Wenn man jedoch fortgeschrittenere Programme erstellt, können sie sehr hilfreich sein. Ein Beispiel hierfür ist die Debugging-Funktion. Diese ermöglicht es, die Ausführung des Programms in einzelne Abschnitte zu zerlegen und dabei die Werte aller Variablen zu betrachten. Das ist sehr hilfreich, wenn man auf der Suche nach einem Fehler im Programm ist. Darüber hinaus ermöglicht es die IDE, vorgefertigte Programmbausteine in den Code einzufügen oder diesen anhand einer grafischen Benutzeroberfläche zu erstellen. Das sorgt bei fortgeschrittenen Programmen für eine einfache und effiziente Gestal-

tungsweise. Aufgrund dieser vielfältigen Vorteile ist die Verwendung von Visual Studio für das Programmieren mit C# sehr zu empfehlen.

2.1 Visual Studio installieren

Bevor man damit beginnt, in C# zu programmieren, ist es daher notwendig, Visual Studio von Microsoft auf dem eigenen Rechner zu installieren. Am sinnvollsten ist es, die Software direkt auf der Homepage des Herstellers herunterzuladen:

https://visualstudio.microsoft.com/de/vs/whatsnew/

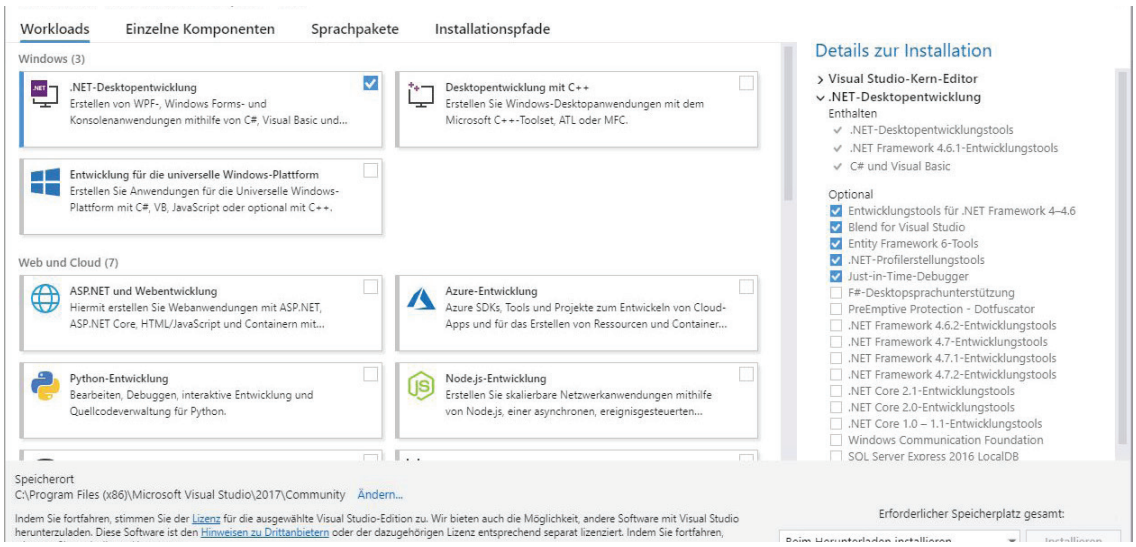
Diese Seite bietet einen Überblick über die Funktionen des Programms und einen Link zum Download der aktuellen Version. Dieses Buch verwendet Visual Studio 2017. Dabei handelt es sich zum Zeitpunkt der Erstellung um die aktuelle Version. Wenn es in der Zwischenzeit in diesem Bereich jedoch zu einer Neuerscheinung gekommen sein sollte, stellt das ebenfalls kein Problem dar. In der Regel werden die grundlegenden Funktionen dabei beibehalten. Es kann jedoch vorkommen, dass es hinsichtlich der grafischen Darstellung einige Unterschiede zu den im Buch abgebildeten Screenshots gibt.

Wenn man mit der Maus über den Download-Button fährt, klappt sich ein Menü auf, das die drei verschiedenen Editionen von Visual Studio anzeigt: Community, Professional und Enterprise. Für diesen Kurs kommt die Community Edition zum Einsatz. Dabei handelt es sich um das einzige der drei Angebote, das kostenfrei erhältlich ist. Die beiden anderen Versionen können nur gegen die Bezahlung einer Lizenz verwendet werden. Dennoch bietet auch die Gratis-Ausführung alle für dieses Buch erforderliche Funktionen und ist für Anfänger vollkommen ausreichend.

Screenshot 2 Die Downloadseite von Visual Studio

Wenn man nach dem Download die Datei anklickt, öffnet sich automatisch der Installations-Assistent. Dieser lädt alle weiteren Dateien herunter und führt den Anwender durch das Setup.

Während der Installation ist es notwendig, vorzugeben, welche Konfiguration gewählt werden soll. Dabei erscheint ein Auswahlfenster mit mehreren Optionen. Für die Bearbeitung der Aufgaben in diesem Buch ist es wichtig, den Bereich .NET-Desktopentwicklung auszuwählen. Alle übrigen Auswahlmöglichkeiten sind für andere Programmiersprachen bestimmt und daher nicht erforderlich.



Screenshot 3 Die .NET-Desktopanwendungen für die Installation auswählen

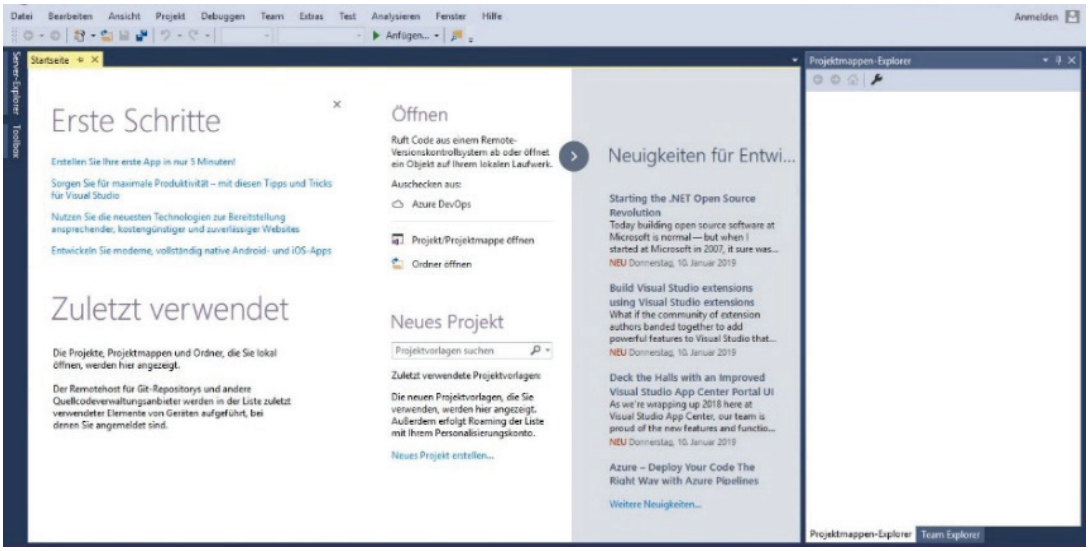
Daraufhin beginnt der eigentliche Installationsprozess, der einige Minuten in Anspruch nehmen kann. Wenn dieser beendet ist, ist das Programm jedoch bereits einsatzbereit.

2.2 Ein neues Projekt mit Visual Studio erstellen

Nun ist es bereits möglich, ein neues Projekt zu erstellen. Dabei soll zunächst lediglich die Funktionsweise von Visual Studio vorgestellt werden, ohne dabei ein Programm zu erzeugen. Dieser Schritt folgt dann im nächsten Kapitel.

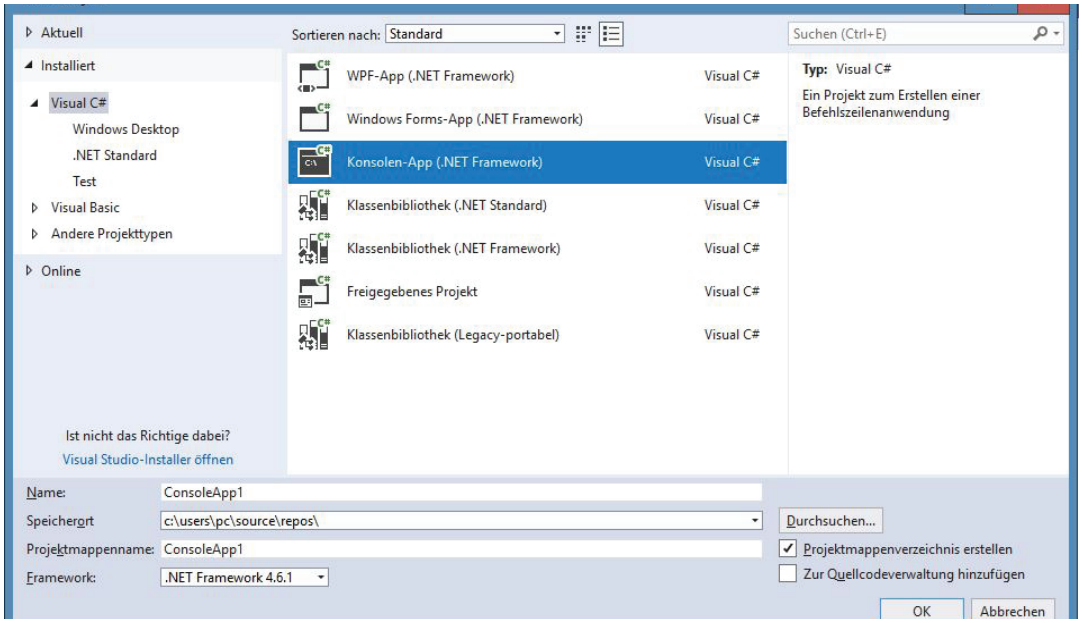
Wenn man Visual Studio nach der Installation das erste Mal öffnet, fragt die Software, welches Farbschema ausgewählt werden soll. Das hat jedoch nur Auswirkungen auf die optische Darstellung. Daher bleibt die Auswahl hierbei jedem Anwender selbst überlassen. Nach dieser Auswahl beginnt die Konfiguration der Software. Das zieht wieder eine kleine Wartezeit nach sich.

Daraufhin öffnet sich das Fenster der IDE mit mehreren Auswahloptionen. Unten in der Mitte befindet sich die Schaltfläche “Neues Projekt erstellen...”. Diese muss angeklickt werden, um das erste Programm zu gestalten.



Screenshot 4 Das Fenster beim Beginn eines neuen Projekts

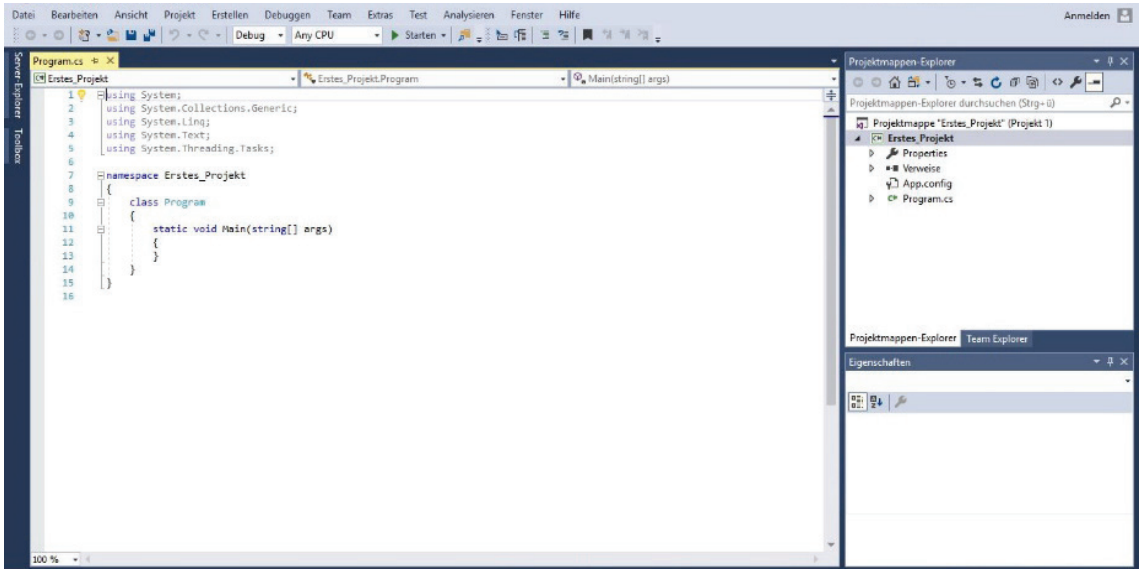
Daraufhin öffnet sich ein neues Fenster. Hier ist es notwendig, die Konsolen-App auszuwählen.



Screenshot 5 Die Erzeugung einer Konsolen-App

Im unteren Bereich ist es möglich, einen Namen für das neue Projekt anzugeben. In diesem Beispiel soll es Erstes_Projekt heißen. Beim Speicherort ist es sinnvoll, die Standardeinstellungen zu übernehmen. Wenn man

nun den OK-Button anklickt, erzeugt Visual Studio die Grundlagen für das erste Programm, das im Rahmen dieses Kurses erstellt werden soll. Daraufhin kehrt die Software wieder zum Hauptfenster zurück. Hier ist nun jedoch ein neuer Bereich entstanden, der bereits einige Codezeilen enthält. Dabei handelt es sich um die grundlegenden Strukturen, die für jedes Programm in C# notwendig sind. Diese haben jedoch keinerlei Funktion. Das Programm lässt sich zwar ausführen, doch passiert dabei zunächst einmal überhaupt nichts. Allerdings sind nun alle Vorbereitungen abgeschlossen, um im nächsten Kapitel mit dem ersten eigenen Programm zu beginnen.



Screenshot 6 Das Fenster für die Bearbeitung des Programmcodes mit den grundlegenden Strukturelementen.

Kapitel 3

Das erste Programm in C#

Nachdem alle Vorbereitungen abgeschlossen sind, ist es an der Zeit, das erste eigene Programm zu erstellen. Dabei handelt es sich um eine sogenannte Konsolenanwendung. Das bedeutet, dass die Ausführung auf der Programmkonsole beruht. Dabei handelt es sich um eine Funktion, die in jedes Betriebssystem bereits integriert ist. Sie ermöglicht es, schriftliche Kommandos einzugeben. Unter Windows trägt das entsprechende Programm den Namen Eingabeaufforderung. Unter Linux ist es unter der Bezeichnung Terminal verfügbar. Konsolen-Anwendungen sind nur in Verbindung mit einem derartigen Programm ausführbar.

Die Konsole kann nur Schrift ausgeben und schriftliche Eingaben entgegennehmen. Eine Konsolenanwendung entspricht daher nicht dem Bild eines modernen Computerprogramms, das über Fenster verfügt und mit der Maus gesteuert wird. Dennoch soll mit dieser Art von Programmen begonnen werden. Der Grund dafür liegt darin, dass sie deutlich einfacher aufgebaut sind. Daher eignen sie sich gut für Anfänger. Erst im weiteren Verlauf dieses Buchs werden dann Programme vorgestellt, die auf Fenstern basieren.

3.1 Eine Ausgabe mit C# erzeugen

Das erste Programm soll eine ganz einfache Aufgabe erfüllen: Es soll einen Text in der Konsole ausgeben. Diese Art von Programm wird auch als Hallo-Welt-Programm bezeichnet. Der Grund dafür liegt darin, dass der erste Schritt beim Erlernen einer Programmiersprache fast immer darin besteht, eine einfache Textausgabe zu erzeugen. In vielen Lehrbüchern ist es üblich, hierfür den Text “Hallo Welt!” beziehungsweise auf Englisch “Hello World!” auszugeben. An dieser Stelle soll das erste Programm jedoch eine Begrüßung zu unserem Kurs erzeugen: “Willkommen zum C#-Kurs!”.

Um das Programm zu schreiben, ist es notwendig, Visual Studio zu öffnen und das Projekt auszuwählen, das bereits im vorherigen Kapitel erstellt wurde. Dabei wurde bereits automatisch etwas Code erzeugt. Das ist sehr hilfreich, da auf diese Weise der Einstieg leichter fällt. Allerdings ist es wich-

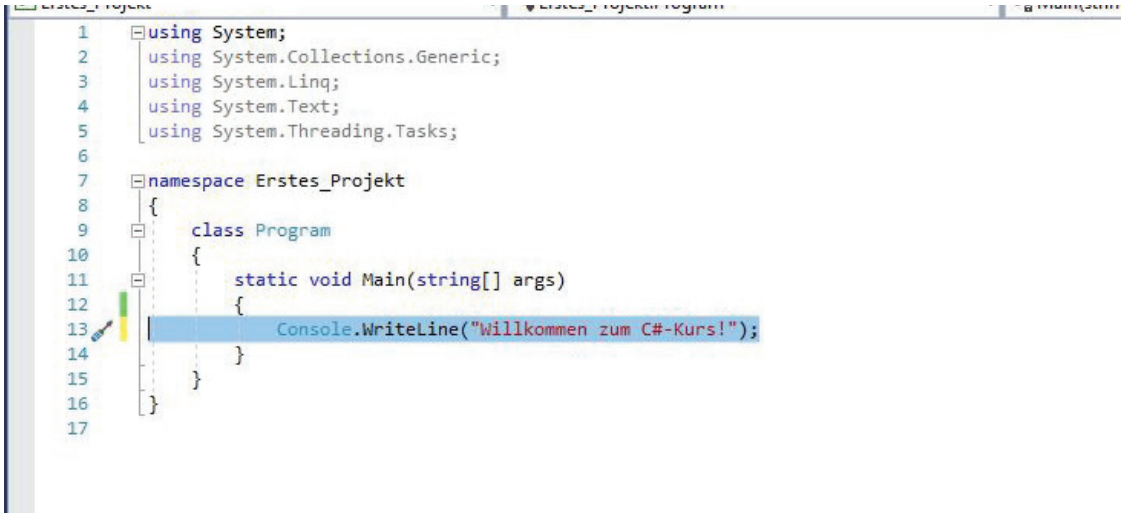
tig, genau darauf zu achten, den eigenen Code an der richtigen Stelle einzufügen. Die bereits vorhandenen Befehle stellen lediglich den Rahmen dar und haben keine Funktion. Sie sind aber für die Ausführung des Programms zwingend notwendig.

Der Einstieg erfolgt in Zeile 12 – nach der sich öffnenden geschweiften Klammer. Der Inhalt des neuen Programms steht stets innerhalb der geschweiften Klammer, die in der folgenden Zeile geschlossen wird. Wenn man mit der Maus hinter die Klammer in Zeile 12 klickt, ist es sinnvoll, erst einmal die Enter-Taste zu betätigen. So entsteht eine neue Zeile für den Code, den man einfügen will.

Dabei kann man bereits eine wichtige Funktion der IDE beobachten: die automatische Einrückung. Der Cursor blinkt nach dieser Aktion noch weiter eingerückt als in der vorherigen Zeile. Der Inhalt der Klammer stellt einen abgeschlossenen Block dar, der eine bestimmte Funktion ausführt. Die IDE macht das deutlich, indem sie den Inhalt automatisch einrückt.

An dieser Stelle soll nun der erste Befehl eingefügt werden:

```
Console.WriteLine("Willkommen zum C#-Kurs!");
```



Screenshot 7 Es ist wichtig, den Befehl an der richtigen Stelle einzufügen.

Dieser Befehl sorgt dafür, dass das Programm die entsprechende Begrüßung über die Konsole ausgibt. Für den Anfänger wirkt diese Ansammlung einzelner Begriffe jedoch recht kompliziert, sodass diese nun Schritt für Schritt erklärt werden sollen.

An erster Stelle steht dabei der Ausdruck Console. Dieser sagt aus, dass sich der folgende Befehl auf die Konsole bezieht – also dass der

Text hier ausgegeben wird. Der zweite Teil des Befehls ist der Begriff `WriteLine`. Dabei handelt es sich um einen Schreib-Befehl, der den entsprechenden Text ausgibt. Darüber hinaus erzeugt er anschließend einen Zeilenumbruch. Alternativ wäre es auch möglich, den einfachen `Write`-Befehl zu verwenden, der keinen Zeilenumbruch erzeugt.

Der Inhalt der Ausgabe muss stets in einer runden Klammer und in Anführungszeichen stehen. Dabei ist es wichtig, zu beachten, dass es sich hierbei nicht um die deutschen Anführungszeichen handelt, sondern um das sogenannte Kodierungszeichen. Wenn man den Code direkt über Visual Studio eingibt, wählt die IDE automatisch das passende Zeichen aus. Wenn man jedoch auf die Idee kommt, den Code zunächst mit einem Textverarbeitungsprogramm zu erstellen und ihn dann in den Texteditor kopiert, kann das zu Problemen führen.

3.2 So ist ein C#-Programm aufgebaut

Im vorherigen Abschnitt wurde nur die Funktionsweise des Schreib-Befehls erklärt, der in den vorgefertigten Programmcode eingefügt wurde. Um die Funktionsweise von C# zu verstehen, ist es jedoch sinnvoll, auch den Code zu betrachten, den Visual Studio automatisch eingefügt hat. Dabei ist es jedoch nicht notwendig, jedes einzelne Detail zu verstehen. Die meisten Befehle werden im weiteren Verlauf des Buchs noch etwas ausführlicher behandelt.

An erster Stelle steht das Kommando `using System`; Dieses ist erforderlich, um die Befehle, die in der Bibliothek `System` enthalten sind, im Programm zu verwenden. C# beinhaltet sehr viele Kommandos. Diese sind jedoch nicht direkt verfügbar, sondern in sogenannten Bibliotheken abgespeichert. Wenn man sie verwenden will, dann ist es notwendig, die entsprechende Bibliothek einzubinden. Der `WriteLine`-Befehl ist beispielsweise in der Bibliothek `System` enthalten. Daher ist er nur verfügbar, wenn man diese zu Beginn in das Programm einbindet.

Danach folgen weitere Befehle für den Import von Bibliotheken. Diese sind für das erste Programm jedoch nicht notwendig, sodass es möglich ist, sie zu löschen, um den Code übersichtlicher zu gestalten.

Anschließend steht das Kommando namespace Erstes_Projekt. Mit C# kann man Funktionen und Objekte erzeugen und diesen einen eigenen Namen geben. Dabei ist es möglich, dass es zu Überschneidungen mit vor-

gefertigten Funktionen kommt. Um dabei eine eindeutige Zuordnung zu gewährleisten, arbeitet C# mit Namensräumen. Innerhalb eines Namensraums darf jede Bezeichnung nur einmal vorkommen. Vor Beginn des Programms ist es notwendig anzugeben, welcher Namensraum verwendet werden soll. Dabei belassen wir es bei der Standardangabe – der Verwendung unseres eigenen Namensraums, der die Bezeichnung Erstes_Projekt trägt.

Danach folgt eine geschweifte Klammer, die deutlich macht, dass nun der eigentliche Inhalt des Programms beginnt. In der nächsten Zeile steht der Begriff `class` Program. Dieser erzeugt eine neue Klasse mit der Bezeichnung Program. Eine Klasse ist das grundlegende Element in der objektorientierten Programmierung. Sie gibt die grundlegenden Strukturen für ein Objekt vor und ermöglicht es, verschiedene Aktionen damit durchzuführen. Die Programmiersprache C# legt viel Wert auf die objektorientierte Programmierung. Das hat zur Folge, dass jedes einzelne Programm die Grundzüge dieses Paradigmas beachten muss. Daher muss jedes Programm als Klasse aufgebaut sein. Deshalb ist es notwendig, es zu Beginn entsprechend zu kennzeichnen und der Klasse einen passenden Namen zu geben. Dieser ist eigentlich frei wählbar. Es ist jedoch sinnvoll, es bei der Standard-Bezeichnung Program zu belassen. Der Inhalt der Klasse steht wiederum in einer geschweiften Klammer. Eine ausführliche Erklärung zu Klassen und Objekten folgt im Kapitel 8.

Danach folgt der Ausdruck `static void Main(string[] args)`. Dieser ist insbesondere für Anfänger schwer zu verstehen. Er ist jedoch von großer Bedeutung, da er den Einstiegspunkt für das Programm darstellt. Wenn man es ausführt, beginnt es stets an dieser Stelle. Dass es sich hierbei um die Hauptfunktion des Programms handelt, wird am Begriff Main deutlich. Davor stehen die Begriffe `static` und `void`. Diese legen einige grundlegende Eigenschaften der Hauptfunktion fest. Die genaue Bedeutung zu erklären, ist jedoch nicht möglich, ohne tiefer auf den Aufbau einer Funktion einzugehen. Daher wird die entsprechende Erklärung im Kapitel 8 nachgereicht, das sich mit diesem Thema beschäftigt.

Zum Abschluss dieser Codezeile steht eine Klammer mit dem Inhalt `string[] args`. Es ist möglich, dem Programm bereits bei dessen Aufruf bestimmte Werte zu übergeben, die Einfluss auf den Ablauf haben. Die Angaben in der Klammer sind erforderlich, um diese Werte innerhalb des Programms aufzunehmen. Obwohl bei diesem Programm keine Übergabe von Werten stattfindet, ist es notwendig, diese Angaben einzufügen.

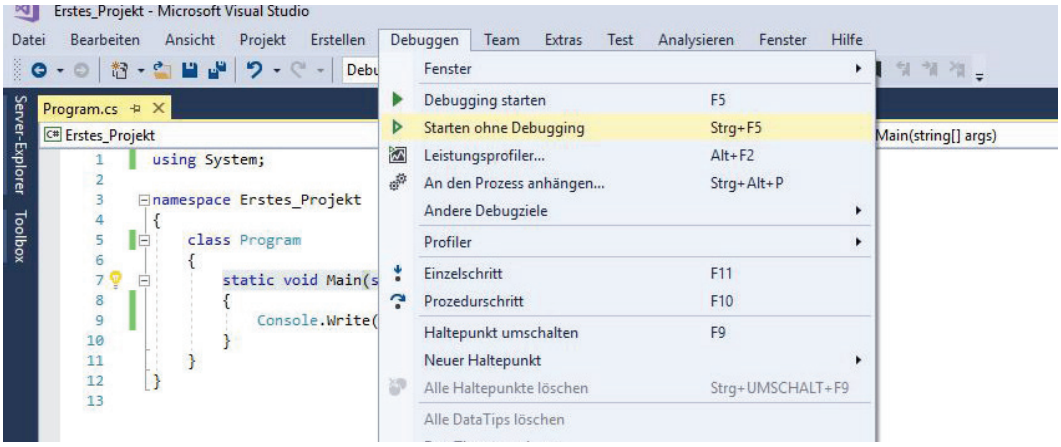
An dieser Stelle ist es sicherlich schwierig, die genaue Funktionsweise der genannten Befehle zu verstehen. Das stellt jedoch kein Problem dar. Wichtig ist es lediglich, darauf zu achten, dass jeder einzelne von ihnen für ein C#-Programm notwendig ist. Wenn man einmal ein Programm mit einem gewöhnlichen Texteditor erstellt, der diesen Code nicht automatisch einfügt, müssen all diese Kommandos von Hand geschrieben werden.

3.3 Den Programmcode kompilieren und ausführen

Das erste Programm ist zwar bereits fertiggestellt und seine Funktionsweise ist erklärt, allerdings wurde es noch nicht ausgeführt. Daher war es bislang nicht möglich, zu überprüfen, welche Auswirkungen es hat.

Bevor man ein C#-Programm ausführen kann, ist es notwendig, es zu kompilieren. Visual Studio macht diese Aufgabe jedoch ganz einfach, indem die IDE die Kompilierung und die Ausführung des Programms in einer Schaltfläche zusammenfasst.

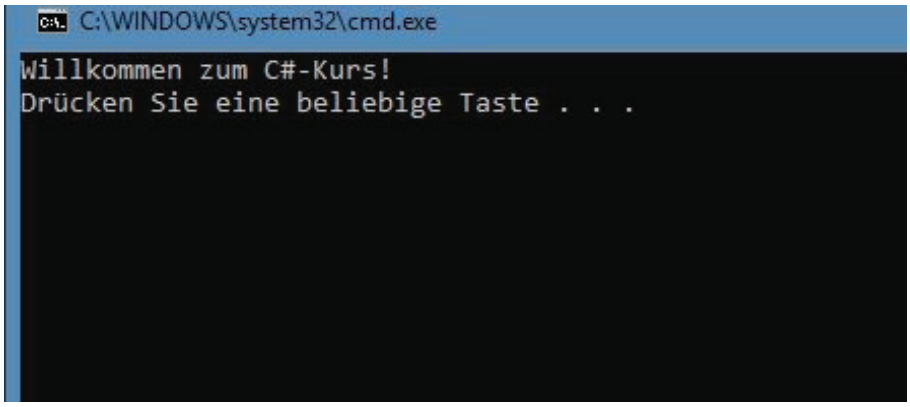
Hierfür ist es lediglich notwendig, in der Menüleiste den Punkt Debuggen auszuwählen und daraufhin "Starten ohne Debugging" auszuwählen.



Screenshot 8 Das Programm kompilieren und ausführen

Alternativ dazu ist es möglich, den Shortcut Strg + F5 zu verwenden. Auf diese Weise lässt sich das Programm in Sekundenbruchteilen ausführen.

Daraufhin öffnet sich die Konsole. Es erscheint der Text, der über den `WriteLine`-Befehl eingegeben wurde. Danach wird automatisch die Zeile "Drücken Sie eine beliebige Taste . . . " eingefügt. Wenn man dieser Aufforderung folgt und eine Taste drückt, schließt sich die Konsole wieder.



Screenshot 9 Die Ausgabe der Begrüßung über die Konsole

Eine weitere Möglichkeit besteht darin, den grünen Pfeil in der Werkzeugleiste mit der Aufschrift Starten anzuklicken. Dieser kompiliert das Programm ebenfalls und führt es aus. Allerdings schließt sich die Konsole dabei sofort wieder, sodass es nicht möglich ist, das Ergebnis zu betrachten.

Die Ausgabe in der Konsole ist nicht der einzige Effekt, der durch die Kompilierung des Programms zu beobachten ist. Darüber hinaus ist es sinnvoll, sich einmal das Verzeichnis anzuschauen, in dem Visual Studio das Projekt abgespeichert hat. Wenn man sich nun durch die Unterverzeichnisse Erstes_Projekt, bin und Debug klickt, entdeckt man eine neue Datei mit dem Namen Erstes_Projekt.exe. Dabei handelt es sich um eine ausführbare Datei, die durch die Kompilierung entstanden ist. Wenn man diese anklickt, öffnet sich ebenfalls kurz die Konsole und zeigt den entsprechenden Text an. Da sie sich jedoch bereits nach Sekundenbruchteilen wieder schließt, ist das kaum zu erkennen. Dennoch ist es wichtig, zu beachten, dass bei jedem Kompilierungs-Prozess eine ausführbare Datei entsteht.

3.4 Kommentare erleichtern das Verständnis des Codes

Wenn man den Programmcode betrachtet, ist es manchmal schwer, zu verstehen, welche Funktionen dieser ausführt. Das ist bereits nicht ganz einfach, wenn man ein eigenes Programm nach einiger Zeit nochmals anschaut. Besonders schwierig ist es, wenn man den Code eines anderen Programmierers liest.

Es gibt jedoch ein praktisches Hilfsmittel, das das Verständnis erleichtert: Kommentare. Dabei handelt es sich um einfachen Text, der keinerlei Auswirkungen auf den Ablauf des Programms hat. Er dient lediglich dazu, die Funktionsweise zu erläutern und auf diese Weise das Verständnis zu erleichtern.

Um deutlich zu machen, dass es sich bei den entsprechenden Textpassagen um Kommentare handelt, die das Programm nicht beeinflussen sollen, ist es notwendig, sie zu kennzeichnen. Für einzeilige Kommentare kommt dafür der doppelte Schrägstrich zum Einsatz (//). Darüber hinaus besteht die Möglichkeit, mehrzeilige Kommentarblöcke einzufügen. Dabei wird der Beginn durch die Symbole /* und das Ende durch die Symbole */ gekennzeichnet. Der folgende Programmcode enthält beide Arten von Kommentaren:

```
using System;

namespace Erstes_Projekt
{
    class Program
    {
        static void Main(string[] args)
        {

            //Dieser Befehl gibt eine Begrüßung aus

            Console.WriteLine("Willkommen zum C#-Kurs!");


                /*
            An dieser Stelle ist das Programm bereits
             beendet.
            Die Konsole wird daraufhin geschlossen.
            */
        }
    }
}
```

Bei einem derart einfachen Programm, ist es sicherlich auch ohne Kommentare möglich, die Funktionsweise zu verstehen. Allerdings ist es sinnvoll, sich diese Vorgehensweise von Beginn an anzugewöhnen. So ist sichergestellt, dass man später bei komplizierteren Programmen die Kommentare automatisch einfügt.

Kommentare sind auch bei der Erstellung eines Programms sehr hilfreich. Wenn man später beispielsweise eine Funktion verwenden will, ist es üblich, zunächst das Hauptprogramm zu schreiben, das diese aufruft. Solange die Funktion noch nicht erstellt ist, lässt sich das Programm jedoch noch nicht ausführen. Um es dennoch auszuprobieren, ist es sinnvoll, den Aufruf der Funktion als Kommentar zu kennzeichnen. So stört er den Ablauf des Programms nicht. Es ist aber dennoch erkenntlich, an welcher Stelle die Funktion später stehen soll. Diese Praxis wird als Auskommentieren bezeichnet.

3.5 Übungsaufgabe: Das erste Programm erweitern

Am Ende der meisten Kapitel steht eine kleine Übungsaufgabe. Diese hat den Zweck, das Erlernte in die Praxis umzusetzen. Auf diese Weise verstärkt sie den Lerneffekt und sorgt für etwas Praxiserfahrung. Sie bezieht sich dabei in erster Linie auf die Inhalte, die im entsprechenden Kapitel vermittelt wurden. Doch auch die Kenntnisse aus den vorherigen Abschnitten werden dabei als gegeben vorausgesetzt. Sollten einmal neue Befehle für die Bearbeitung notwendig sein, werden diese kurz erläutert.

Danach folgt eine Musterlösung für die entsprechenden Aufgaben. Es wird jedoch dringend empfohlen, zunächst selbst zu probieren, ein entsprechendes Programm zu erstellen – selbst wenn das beim ersten Anlauf nicht gelingen sollte. Die Lösung dient in erster Linie zum Abgleich mit den eigenen Ergebnissen. Erst wenn das Programm auch nach mehreren Versuchen seine Funktion nicht erfüllt, ist es sinnvoll, hier nachzuschauen.

Beim Erstellen eines Programms gibt es meistens viele verschiedene Lösungswege. Das bedeutet, dass es sich bei der Musterlösung nicht um die einzige Möglichkeit handelt. Wenn das eigene Programm ausführbar ist und alle Anforderungen erfüllt, dann stellt es ebenfalls eine richtige Lösung dar, auch wenn es von der Musterlösung abweicht.

Da in diesem Kapitel nur ein einziger Befehl vorgestellt wurde, ist die erste Übungsaufgabe selbstverständlich sehr einfach. Sie dient in diesem Fall lediglich der Wiederholung und der Vertiefung der erworbenen Kenntnisse.

1. Erstellen Sie ein Programm, das eine weitere Zeile zur Ausgabe hinzufügt, die auf den Inhalt des Kurses eingeht.
2. Fügen Sie auch für diese zweite Zeile einen passenden Kommentar ein.

Lösungen:

1.

```
using System;

namespace Erstes_Projekt
{
    class Program
    {
        static void Main(string[] args)
        {
            //Dieser Befehl gibt eine Begrüßung aus
            Console.WriteLine("Willkommen zum C#-Kurs!");
            Console.WriteLine("Hier lernen Sie, Programme mit
             C# zu erstellen.");
        }
    }
}
```

2.

```
using System;

namespace Erstes_Projekt
{
    class Program
    {
        static void Main(string[] args)
        {
            //Dieser Befehl gibt eine Begrüßung aus
            Console.WriteLine("Willkommen zum C#-Kurs!");
            //Diese Ausgabe geht auf die Inhalte ein.
            Console.WriteLine("Hier lernen Sie, Programme mit
             C# zu erstellen.");
        }
    }
}
```

Hat Ihnen diese Leseprobe gefallen?

Bestellen Sie das Buch als Taschenbuch oder eBook komfortabel auf Amazon!

eBook: https://amzn.to/2T3jRTt

Taschenbuch: https://amzn.to/2SosaE1

Möchten Sie über neue Bücher und Sonderangebote vom BMU Verlag informiert werden?

Tragen Sie sich jetzt in unseren E-Mail Newsletter ein: https://bmu-verlag.de/