HANS-GEORG SCHUMANN

C# FÜR KIDS

PROGRAMMIEREN LERNEN OHNE VORKENNTNISSE

mitp

Hans-Georg Schumann

C# FÜR KIDS

PROGRAMMIEREN LERNEN OHNE VORKENNTNISSE

mitp

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Bei der Herstellung des Werkes haben wir uns zukunftsbewusst für umweltverträgliche und wiederverwertbare Materialien entschieden.
Der Inhalt ist auf elementar chlorfreiem Papier gedruckt.

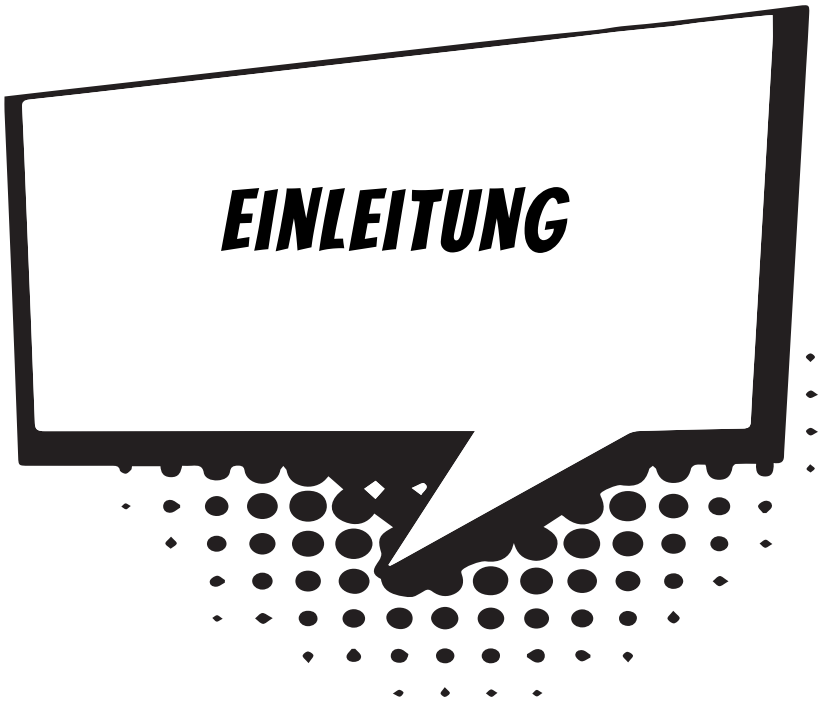
ISBN 978-3-7475-0587-8
1. Auflage 2022

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Katja Völpel
Sprachkorrektorat: Petra Heubach-Erdmann
Covergestaltung: Christian Kalkert
Satz: Satz: III-satz, Kiel, www.drei-satz.de

INHALT

Für
Janne, Julia, Katrin und Daniel



EINLEITUNG

Computer und Smartphones sind schon wahre Wunder-Maschinen. Doch das sind sie nur dadurch, dass es schlaue Programmierer gibt, die die passenden Anwendungen und Spiele erstellen. Wenn du eines Tages zur Gilde der Programmierer zählen willst, dann hast du hier die Möglichkeit, bei null anzufangen – ohne irgendwelche Programmierkenntnisse.

Aber warum selbst Programme erstellen, wo es doch zahlreiche Spiele und Apps schon gibt? Wenn du ehrlich bist, wirst auch du zugeben müssen: Nicht immer gefällt einem das Angebot, das es gibt, man hätte schon gern das eine oder andere anders gehabt. Oder gemacht. Und das kann man nur, wenn man programmieren kann.

Programmiersprachen gibt es reichlich. Und da ist es für einen Einsteiger nicht leicht, sich eine auszusuchen. Hier lernst du, in C# zu programmieren; das ist eine sehr mächtige und leistungsfähige Sprache, nicht kinderleicht zu erlernen, aber die Mühe lohnt sich auf jeden Fall.

WAS HEISST EIGENTLICH PROGRAMMIEREN?

Wenn du aufschreibst, was ein Computer tun soll, nennt man das **Programmieren**. Das Tolle daran ist, dass du selbst bestimmen kannst, was getan werden soll. Lässt du dein Programm laufen, macht der Computer die Sachen, die du ausgeheckt hast. Natürlich wird er dann nicht dein Zimmer aufräumen und dir auch keine Tasse Kakao ans Bett bringen. Aber kannst du erst mal programmieren, kannst du den Computer sozusagen nach deiner Pfeife tanzen lassen.

Allerdings passiert es gerade beim Programmieren, dass der Computer nicht so will, wie du es gerne hättest. Meistens sind das Fehler im Programm. Das Problem kann aber auch irgendwo anders im Computer oder im Betriebssystem liegen. Das Dumme bei Fehlern ist, dass sie sich gern so gut verstecken, dass die Suche danach schon manchen Programmierer zur Verzweiflung gebracht hat.

Vielleicht hast du nun trotzdem Lust bekommen, das Programmieren zu erlernen. Dann brauchst du ja nur noch eine passende **Entwicklungsumgebung**, und schon kann's losgehen.

WAS IST EINE ENTWICKLUNGSUMGEBUNG?

Um ein Programm zu erstellen, musst du erst mal etwas eintippen. Das ist wie bei einem Brief oder einer Geschichte, die man schreibt. Das Textprogramm dafür kann sehr einfach sein, weil es ja nicht auf eine besondere Schrift oder Darstellung ankommt wie bei einem Brief oder einem Referat. So etwas wird **Editor** genannt.

Ist das Programm eingetippt, kann es der Computer nicht einfach so lesen und ausführen. Jetzt muss es so übersetzt werden, dass der PC versteht, was du von ihm willst. Weil er aber eine ganz andere Sprache spricht als du, muss ein Dolmetscher her.

Du programmierst in einer Sprache, die du verstehst, und der Dolmetscher übersetzt es so, dass es dem Computer verständlich wird. So etwas heißt dann **Compiler**. Schließlich müssen Programme getestet, überarbeitet, verbessert, wieder getestet und weiterentwickelt werden. Dazu gibt es noch einige zusätzliche Hilfen. Daraus wird dann ein ganzes System, die Entwicklungsumgebung.

WARUM GERADE C#?

Leider kannst du nicht so programmieren, wie dir der Mund gewachsen ist. Eine Programmiersprache muss so aufgebaut sein, dass möglichst viele Menschen in möglichst vielen Ländern einheitlich damit umgehen können.

Weil in der ganzen Welt Leute zu finden sind, die wenigstens ein paar Brocken Englisch können, besteht auch fast jede Programmiersprache aus englischen Wörtern. Es gab auch immer mal Versuche, z.B. in Deutsch zu programmieren, aber meistens klingen die Wörter dort so künstlich, dass man lieber wieder aufs Englische zurückgreift.

Eigentlich ist es egal, welche Programmiersprache du benutzt. Am besten eine, die möglichst leicht zu erlernen ist. Aber sie soll auch leistungsfähig sein. In diesem Buch hast du es mit der Programmiersprache C# zu tun. Sie ist weit verbreitet, ist nicht einfach, aber auch für Anfänger geeignet, die zum ersten Mal programmieren lernen wollen. (Willst du mal in andere Sprachen hineinschnuppern, dann empfehle ich dir z.B. eines der »... für Kids«-Bücher über C++, Java, JavaScript oder Python.)

Der Weg zum guten Programmierer kann ganz schön steinig sein. Nicht selten kommt es vor, dass man die Lust verliert, weil einfach nichts klappen will. Das Programm tut etwas ganz anderes, man kann den Fehler nicht finden und man fragt sich: Wozu soll ich eigentlich programmieren lernen, wo es doch schon genug Programme gibt? Und dann noch ausgerechnet in C#.

Immer wieder werden gute Programmierer dringend gesucht, und dieser Bedarf wird weiter steigen. Wirklich gute Programmierer werden auch wirklich gut bezahlt. Es ist also nicht nur einen Versuch wert, es kann sich durchaus lohnen, das Programmieren in C# zu erlernen.

VISUAL STUDIO, DIE ENTWICKLUNGSUMGEBUNG ZUM BUCH

Um den Kauf einer Entwicklungsumgebung für C# musst du dich nicht weiter kümmern, denn die bekommst du kostenlos aus dem Internet. Mit einer kostenlosen Version der Software **Visual Studio** von Microsoft hast du eine weitverbreitete Entwicklungsumgebung und kannst damit unter allen Versionen von Windows programmieren.

Das komplette Paket findest du auf dieser Seite:

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

UND WAS BIETET DIESES BUCH?

Über eine ganze Reihe von Kapiteln verteilt lernst du

- die Grundlagen von C# kennen
- mit Visual Studio unter Windows umzugehen

- einiges über die objektorientierte Programmierung (OOP)
- mit Komponenten zu arbeiten (das sind Bausteine, mit denen du dir viel Programmierarbeit sparen kannst)
- die grafischen Möglichkeiten von C# kennen
- eine Reihe von Spielen selbst zu programmieren

Im **Anhang** gibt es dann noch einiges an Informationen und Hilfen, u.a. über Installationen und den Umgang mit Fehlern.

WIE ARBEITEST DU MIT DIESEM BUCH?

Grundsätzlich besteht dieses Buch aus einer Menge Text mit vielen Abbildungen dazwischen. Natürlich habe ich mich bemüht, alles so zuzubereiten, dass daraus lauter gut verdauliche Happen werden. Damit das Ganze noch genießbarer wird, gibt es zusätzlich noch einige Symbole, die ich dir hier gern erklären möchte:

ARBEITSSCHRITTE

≫ Wenn du dieses Zeichen siehst, heißt das: Es gibt etwas zu tun. Damit kommen wir beim Programmieren Schritt für Schritt einem neuen Ziel immer näher.

Grundsätzlich lernt man besser, wenn man einen Programmtext selbst eintippt oder ändert. Aber nicht immer hat man große Lust dazu. Deshalb gibt es alle Projekte im Buch auch als Download:

https://www.mitp.de/0586

Und hinter einem Programmierschritt findest du auch den jeweiligen Namen des Projekts oder einer Datei (z.B. PROJEKT1, GRAFIK1, GAME1). Wenn du also das Projekt nicht selbst erstellen willst, kannst du stattdessen diese Datei laden (zu finden im Ordner PROJEKTE).

AUFGABEN

Am Ende eines Kapitels gibt es meist eine Reihe von Fragen und Aufgaben. Diese Übungen sind nicht immer ganz einfach, aber sie helfen dir, noch besser zu programmieren. Lösungen zu den Aufgaben findest du in verschiedenen Formaten ebenfalls bei den Download-Dateien. Du kannst sie dir alle im Editor von Windows oder auch in deinem Textverarbeitungsprogramm anschauen. Oder du lässt sie dir ausdrucken und hast sie dann schwarz auf weiß, um sie neben deinen PC zu legen. (Auch die Programme zu den Aufgaben liegen im Ordner PROJEKTE.)

NOTFÄLLE UND DIREKTHILFE

Vielleicht hast du irgendetwas falsch gemacht oder etwas vergessen. Oder es wird gerade knifflig. Dann fragst du dich, was du nun tun sollst. Bei diesem Symbol findest du eine Lösungsmöglichkeit. Es kann nicht schaden, auch mal ganz hinten im Anhang B nachzuschauen, wo ein paar Hinweise zur Pannenhilfe aufgeführt sind.

ACHTUNG

Hin und wieder findest du ein solch dickes Ausrufezeichen im Buch. Dann ist das eine Stelle, an der etwas besonders Wichtiges steht.

SPEZIALWISSEN

Wenn du ein solches »Wow« siehst, geht es um ausführlichere Informationen zu einem Thema.

WAS BRAUCHST DU FÜR DIESES BUCH?

Du findest Visual Studio als komplette Entwicklungsumgebung zum Download auf dieser Homepage:

https://www.visualstudio.com/de/downloads/

Visual Studio ist in der **Community**-Version kostenlos, und die reicht komplett aus, um mit C# auch umfangreiche Programme zu erstellen. Nach dem Download wird alles mit dem **Setup**-Programm in ein Verzeichnis deiner Wahl installiert, z.B. c:\Programme\MICROSOFTVisual Studio.

Für deine C#-Projekte solltest du einen Extra-Ordner benutzen. Die Beispielprogramme in diesem Buch gibt es alle als Download von der Homepage des Verlages, falls du mal keine Lust zum Abtippen hast:

https://www.mitp.de/0586

Und auch die Lösungen zu den Fragen und Aufgaben sind dort untergebracht.

BETRIEBSSYSTEM

Die meisten Computer arbeiten heute mit dem Betriebssystem Windows. Davon brauchst du eine der Versionen 7 bis 11.

SPEICHERMEDIEN

Auch wenn du deine Programme auf der Festplatte unterbringst, kann es nicht schaden, sie zusätzlich z.B. auf einem USB-Stick als Backup zu speichern.

Gegebenenfalls bitte deine Eltern oder Lehrer um Hilfe.

HINWEISE FÜR LEHRER

Dieses Buch versteht sich auch als Lernwerk für den Informatik-Unterricht in der Schule. Dort setzt natürlich jeder Lehrer seine eigenen Schwerpunkte. Benutzen Sie an Ihrer Schule bereits ein Werk aus einem Schulbuchverlag, so lässt sich dieses Buch auch als Materialienband einsetzen – in Ergänzung zu dem vorhandenen Schulbuch. Weil dieses Buch sozusagen »von null« anfängt, ist ein direkter Einstieg in C# möglich – ohne irgendwelche anderen Programmierkenntnisse.

Ein wichtiger Schwerpunkt in diesem Buch ist die objektorientierte Programmierung (OOP). Auf die elementaren Eigenheiten (Kapselung, Vererbung und Polymorphie) wird ausführlich eingegangen.

In den Projekten werden alle wesentlichen Elemente des C#-Wortschatzes wie auch die wichtigsten Grafik-Komponenten eingesetzt. Ein besonderer Schwerpunkt liegt auf der Spieleprogrammierung.

In den Lösungen zu den Aufgaben finden Sie weitere Vorschläge zur Programmierung in C#.

ÜBUNGSMEDIEN

Für den Informatik-Unterricht sollte jeder Schüler ein anderes externes Speichermedium haben, um darauf seine Programmierversuche zu sichern. So wird verhindert, dass sich auf der Festplatte des Schulcomputers mit der Zeit allerlei »Datenmüll« ansammelt. Außerdem dient der eigene Datenträger dem Datenschutz: Nur der betreffende Schüler kann seine Daten manipulieren.

REGELMÄßIG SICHERN

Es kann nicht schaden, die Programmdateien, an denen gerade gearbeitet wird, etwa alle zehn Minuten zu speichern. Denn Computer pflegen gern gerade dann »abzustürzen«, wenn man seine Arbeit längere Zeit nicht gespeichert hat.

Das ist aber nur dann nötig, wenn man ein Programm längere Zeit nicht startet. In der Regel fragt nämlich Visual Studio bei jedem Programmstart nach, ob die Datei gespeichert werden soll.

1 ERSTE SCHRITTE MIT C#

Am besten legen wir gleich los mit dem Programmieren. Nach dem Start des Computers und dem Auftauchen von Windows können wir uns schon dem ersten C#-Projekt widmen. Es wird natürlich noch kein Computerspiel sein, aber es gibt schon einiges zum Herumspielen.

In diesem Kapitel lernst du

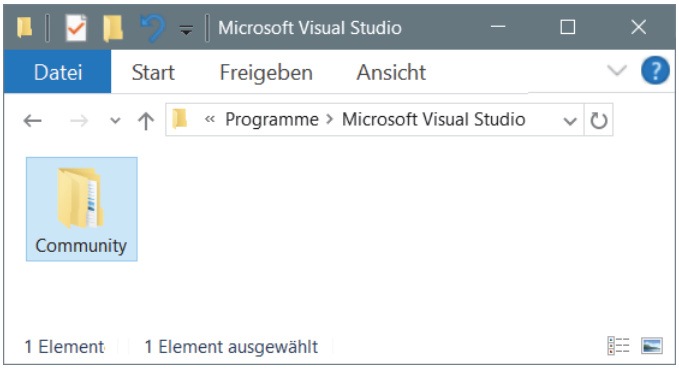
- wie man Visual Studio startet
- wie man ein Programm erstellt und ausführt
- einiges über (mögliche) Fehler
- was `WriteLine()` und `ReadLine()` bedeuten
- etwas über Variablen
- wie man ein Projekt speichert
- wie man Visual Studio beendet

VISUAL STUDIO STARTEN

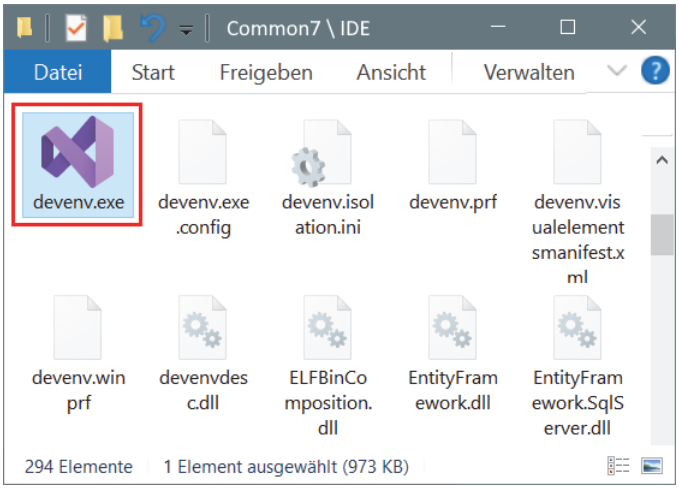
Bevor wir mit dem Programmieren anfangen können, muss **Visual Studio** erst installiert werden. Die Installation übernimmt ein Programm namens SETUP. Genaues erfährst du im Anhang A. Hier musst du dir von jemandem helfen lassen, wenn du dir die Installation nicht allein zutraust.

Eine Möglichkeit, Visual Studio zu starten, ist diese:

≫ Öffne den Ordner, in dem du Visual Studio untergebracht hast (z.B. C:\PROGRAMME\ MICROSOFT VISUAL STUDIO).

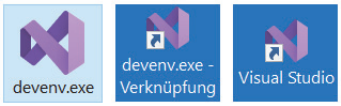


≫ Dort musst du nun weiter über einige Unterordner wie COMMUNITY und COMMON7 in IDE wechseln:



≫ Hier suchst du unter den zahlreichen Symbolen eines derjenigen heraus, bei denen etwas aussieht wie eine gekippte lila 8, und zwar das mit dem Namen DEVENV.EXE. Das Symbol wird nicht so schnell zu finden sein.

≫ Doch dann kannst du das Programm mit einem Doppelklick auf das Symbol starten:

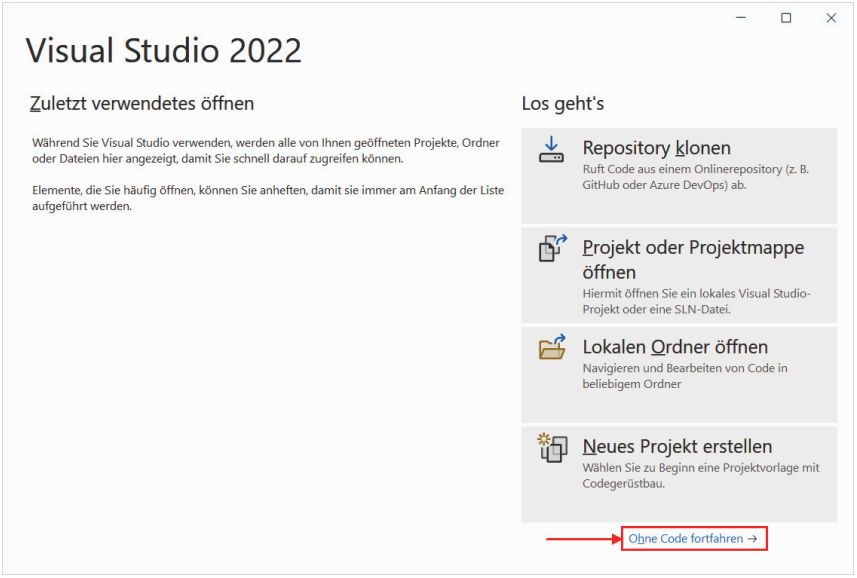


Ich empfehle dir, eine **Verknüpfung** auf dem Desktop anzulegen:

- Dazu klickst du mit der rechten Maustaste auf das Symbol für Visual Studio (DEVENV.EXE). Im Kontextmenü wählst du KOPIEREN.
- Dann klicke auf eine freie Stelle auf dem Desktop, ebenfalls mit der rechten Maustaste. Im Kontextmenü wählst du VERKNÜPFUNG EINFÜGEN.
- Es ist sinnvoll, für das neue Symbol auf dem Desktop den Text DEVENV.EXE – VERKNÜPFUNG durch VISUAL STUDIO zu ersetzen.

Von nun an kannst du auf das neue Symbol auf dem Desktop **doppelklicken** und damit Visual Studio starten.

Je nach Computer kann es eine Weile dauern, bis Visual Studio geladen ist. Was dich schließlich erwartet, könnte ungefähr so aussehen:



Wenn du hier auf NEUES PROJEKT ERSTELLEN klickst, beginnt dein erstes Projekt. Du kannst also gleich loslegen, wenn du willst. Oder:

≫ Du klickst erst einmal auf OHNE CODE FORTFAHREN, um im Hauptfenster von Visual Studio zu landen.

Dort schauen wir uns jetzt ein bisschen um. Sieht ganz schön leer aus, doch uns soll jetzt nur die Menüleiste interessieren – ganz oben:

Links darunter befinden sich jede Menge Symbole, die man mit der Maus anklicken kann. Zu einigen davon kommen wir im Laufe der folgenden Kapitel noch.

Diese Menüs von Visual Studio wirst du wahrscheinlich am meisten benutzen:

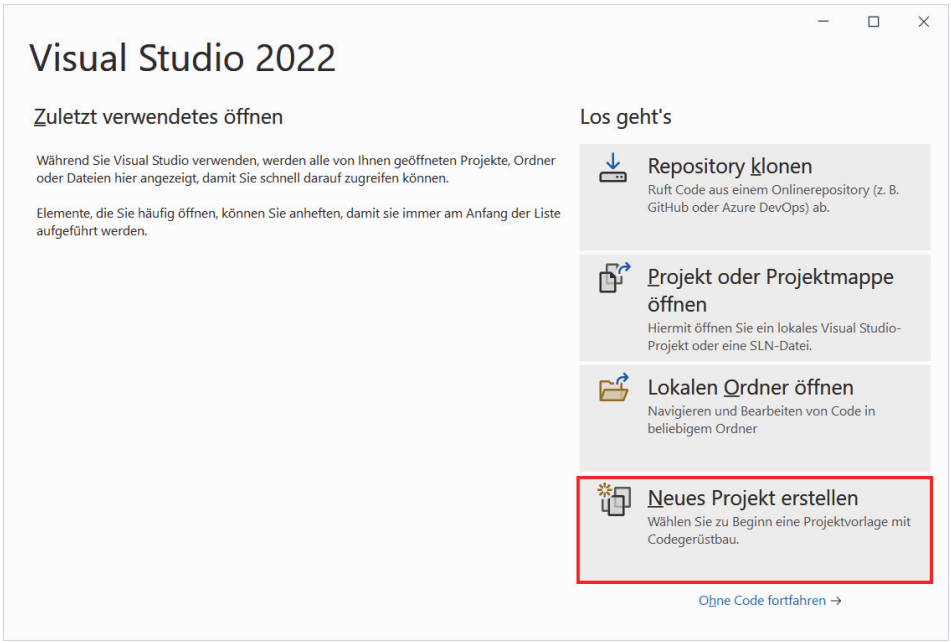
- Über das DATEI-Menü kannst du Dateien speichern, laden (öffnen), ausdrucken, neu erstellen oder Visual Studio beenden.
- Das BEARBEITEN-Menü hilft dir bei der Bearbeitung deines Programmtextes, aber auch bei anderen Programmelementen. Außerdem kannst du dort bestimmte Arbeitsschritte rückgängig machen oder wiederherstellen.
- Im ANSICHT-Menü hast du unter anderem die Möglichkeit, zusätzliche Hilfsfenster und Boxen ein- oder auszublenden.
- Über das DEBUGGEN-Menü sorgst du dafür, dass dein Programmprojekt ausgeführt wird.
- Und das HILFE-Menü bietet dir vielfältige Hilfe-Informationen an.

Einige wichtige Menüeinträge sind in einem sogenannten **Popup**-Menü zusammengefasst. Das heißt so, weil es dort aufklappt, wo du gerade mit der **rechten** Maustaste hinklickst.

Ein Editorfenster, wie du es vielleicht von einem Editor oder Textverarbeitungsprogramm her kennst, ist gerade nicht in Sicht. Aber das lässt sich ändern: Ziel ist es ja, ein neues Projekt – unser Erstlingswerk – zu erstellen. Also los!

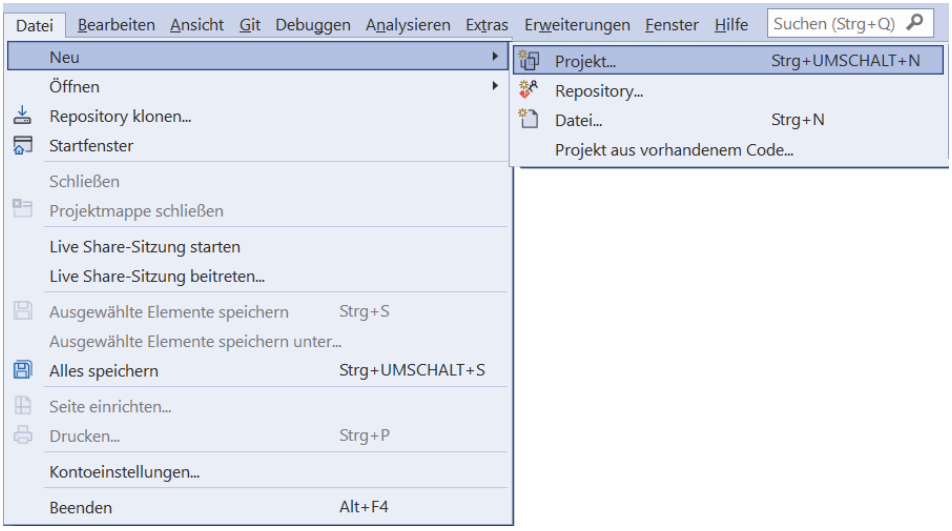
DAS ERSTE PROGRAMM

≫ Es gibt nun diese zwei Wege. Entweder du schließt das Fenster von Visual Studio (Klick auf das X rechts oben) und startest es neu – womit du in diesem Fenster landest:

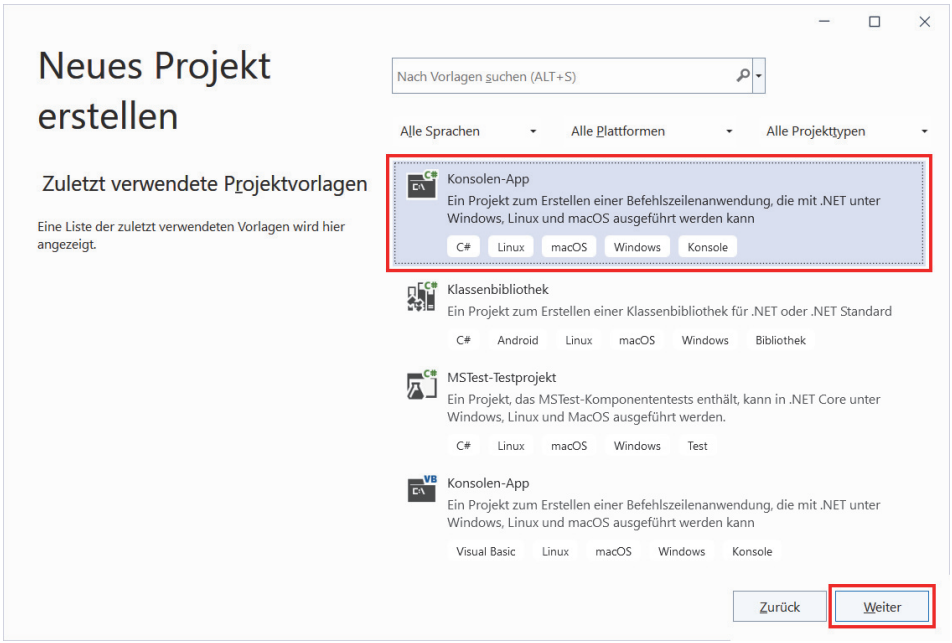


≫ Dann klickst du auf NEUES PROJEKT ERSTELLEN.

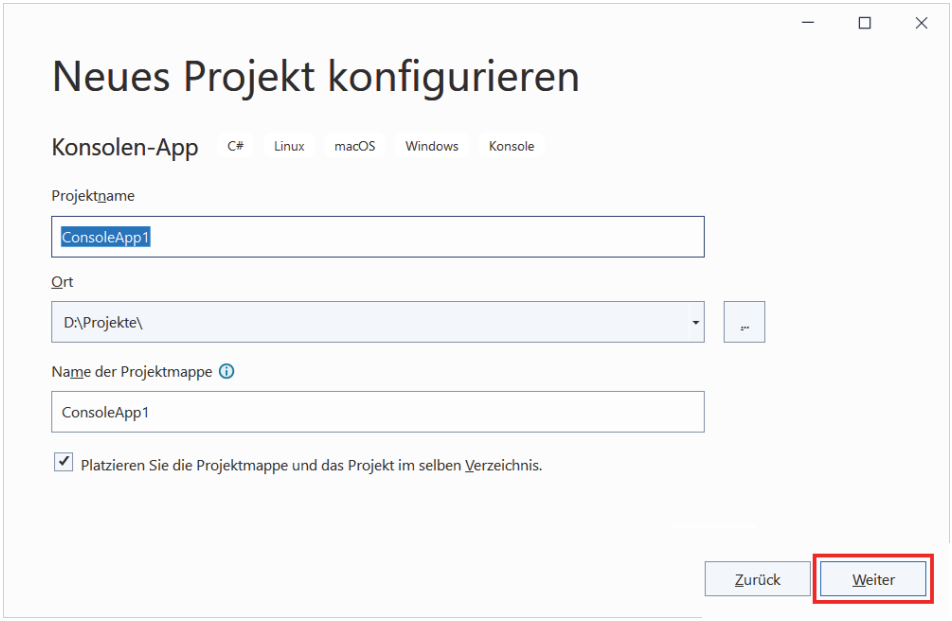
≫ Oder du hast dich entschieden, bei der Menüleiste von Visual Studio zu bleiben. Dann klickst du dort auf DATEI und im sich öffnenden Menü auf NEU und dann auf PROJEKT.



Es erscheint ein Dialogfeld, in dem es offenbar mehrere Möglichkeiten gibt, wie du dein Projekt erstellst:



≫ Wähle rechts **oben** die Option KONSOLEN-APP (dort findest du auch den Eintrag C#). Dann klicke unten rechts auf WEITER.



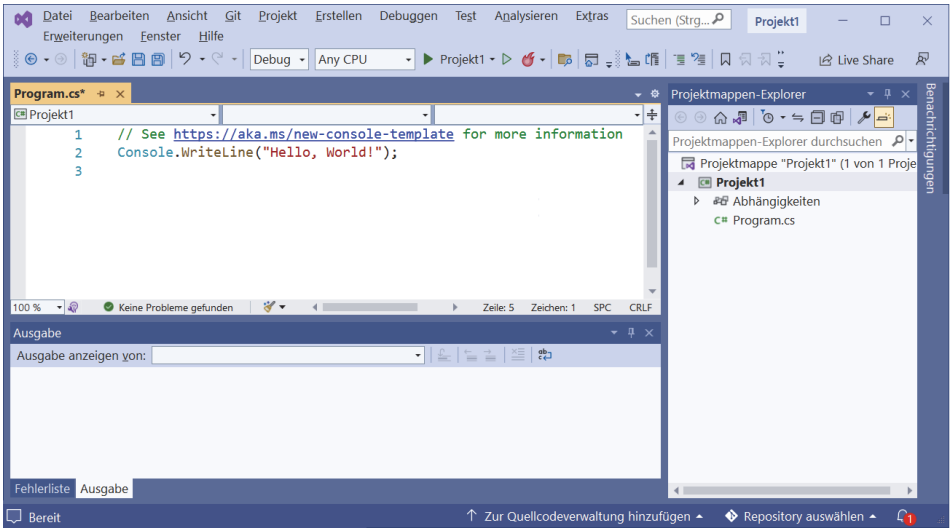
≫ Gib dem Projekt einen Namen und sorge dafür, dass hinter SPEICHERORT der Pfad steht, wo du dein Projekt unterbringen willst. Dann klicke abschließend auf WEITER.

Ich habe das Ganze einfallslos erst mal PROJEKT1 genannt. Du kannst den vorgegebenen Speicherort übernehmen, ich empfehle dir aber, besser einen eigenen Ordner zu erzeugen und den dort anzugeben. Bei mir ist das der Ordner PROJEKTE auf Laufwerk D:.

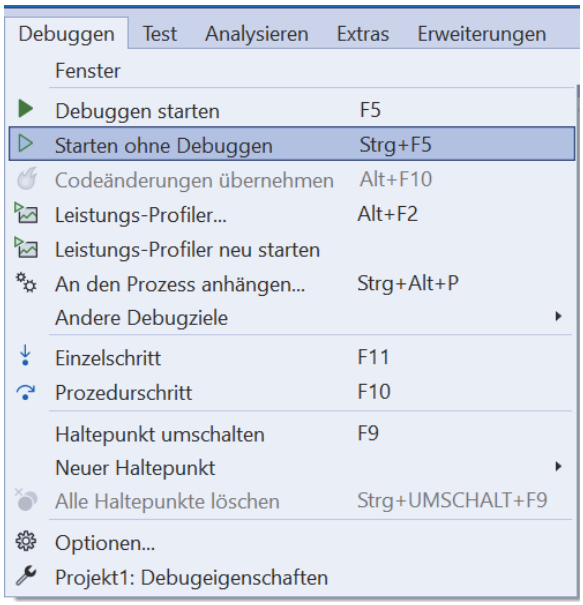
≫ Im nächsten Dialogfeld erzeugst du mit einem abschließenden Klick auf ERSTELLEN dein erstes Mini-Projekt in C#.

DER QUELLTEXT

Wie du im linken großen Fensterbereich sehen kannst, bietet Visual Studio schon ein kleines Gerüst-Programm, das du natürlich auch starten kannst, allerdings passiert noch nichts Aufregendes.

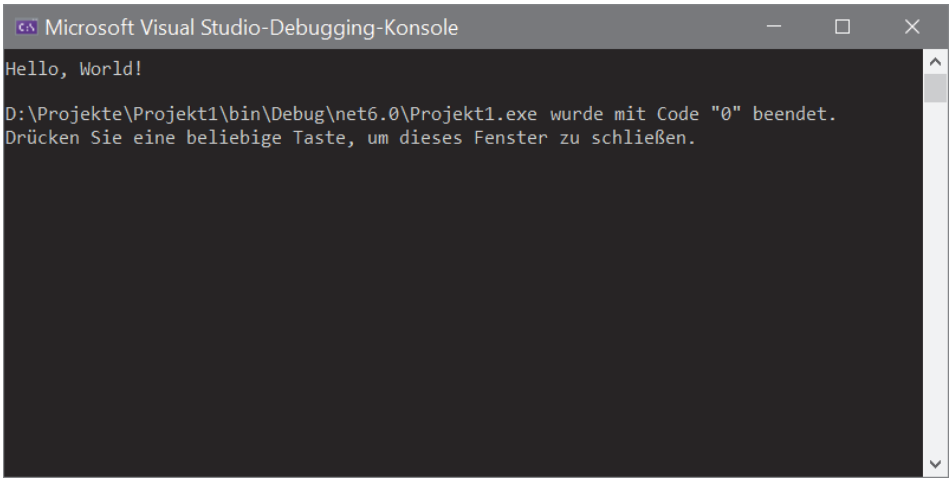


≫ Probiere ruhig mal aus, was sich tut, wenn du in der Menüleiste auf DEBUGGEN und STARTEN OHNE DEBUGGEN klickst. (Alternativ kannst du auch die Tastenkombination Strg + F5 drücken.)



Und nach dem Start des Programms tut sich tatsächlich etwas: Unten im Ausgabefenster steht einiges an Meldungen, die du jetzt einfach ignorieren solltest. Wichtiger ist das neue Fenster mit schwarzem Hintergrund, das sich öffnet und sich vor Visual Studio schiebt. Das ist das Konsolenfenster. Und in dem steht zum einen ein kurzer Gruß, darunter dann etwas momentan vielleicht für dich unverständliches

Kauderwelsch. Wichtig ist der letzte Satz, die Aufforderung, eine beliebige Taste zu drücken.



≫ Folge dieser Aufforderung (oder Bitte), um das Fenster wieder zu schließen.

Na ja, überwältigend war deine erste Begegnung mit einem C#-Programm nicht, aber es liegt an dir, daraus mehr zu machen. Zuerst aber schauen wir uns jetzt das näher an, was da links oben im Editorfenster steht:

```
// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, World!");
```

Das, was du da siehst, wird **Quelltext** genannt. Die erste Zeile mit den beiden Schrägstrichen (//) am Anfang ist ein **Kommentar**. Der weist in diesem Falle darauf hin, dass du mehr unter der angezeigten Internet-Adresse erfahren kannst.

Als Kommentar könnte man aber auch so etwas hinschreiben:

```
// Mein erstes Projekt
```

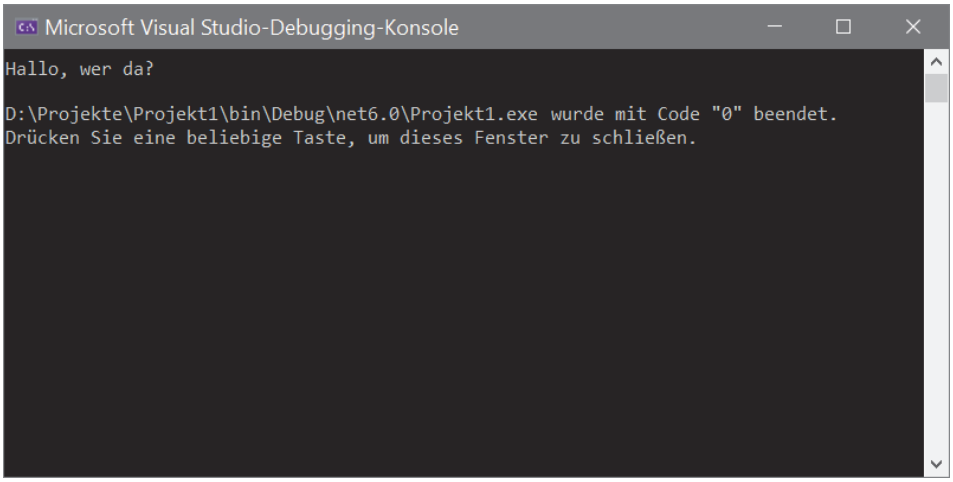
Oder man lässt diesen Kommentar einfach ganz weg. Für ein funktionierendes Programm ist nur diese Zeile nötig:

```
Console.WriteLine("Hello, World!");
```

Und da steht schon einiges, was für uns interessant ist. Offenbar bewirkt diese Zeile, dass ein bestimmter Text ausgegeben wird. In diesem Fall ist es »Hello, World«, aber das lässt sich leicht ändern:

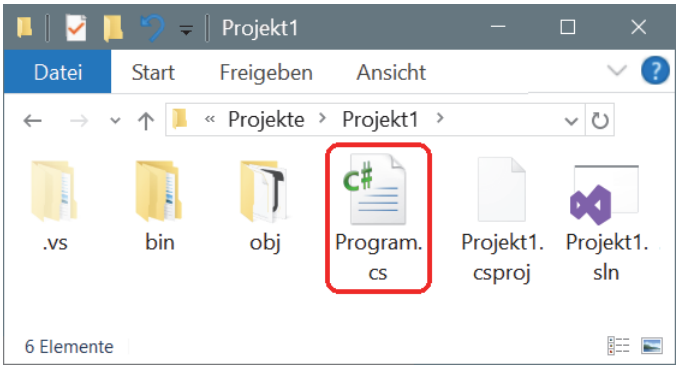
```
Console.WriteLine("Hallo, wer da?");
```

≫ Passe den Programmtext an, indem du einiges löschst und ersetzt. Dann starte das Programm erneut über DEBUGGEN und STARTEN OHNE DEBUGGINGoder mit Strg + F5.



Und wie du siehst, zeigt das Programm nun **deinen** Text an.

Das soll alles sein? Schauen wir mal im PROJEKTE-Ordner nach. Dort liegt ein Unterordner mit dem Namen PROJEKT1. Wenn du den öffnest, erwartet dich ein solches Bild:



Da liegt ja einiges an Dateien herum und in den Unterordnern findest du noch mehr. All das hat Visual Studio automatisch erzeugt, es muss dich aber nicht weiter interessieren, denn die Datei, um die es für **dich** geht, heißt PROGRAM.CS.

»CS« ist die Kennung für Dateien mit C#-Quelltext und kürzt »C-sharp« ab, so spricht man C# auch aus. Wenn du diese Datei mit einem normalen Texteditor öffnest, findest du darin die obigen Programmzeilen.

Und nun klären wir erst einmal, was diese eine Textzeile, aus der unser Programm besteht, bedeutet. Den Kommentar darüber kann man entsorgen, also löschen. Oder du trägst hinter die zwei Schrägstriche einen eigenen Infotext ein.

Es handelt sich hier um eine **Anweisung**, die den Computer dazu bringt, dich freundlicher zu grüßen:

```
Console.WriteLine("Hallo, wer da?");
```

Fangen wir bei dieser Anweisung von hinten an. Dort steht der Text, der angezeigt werden soll, eingepackt in Anführungsstriche und zusätzlich in runden Klammern:

```
("Hallo, wer da?")
```

Für die Anzeige sorgt die Anweisung `WriteLine()`, was ausführlich heißt: »Schreib etwas auf dem Bildschirm und gehe danach in die nächste Zeile.«

Es ist übrigens nicht egal, ob für die Wörter große oder kleine Buchstaben benutzt werden. C# unterscheidet eindeutig zwischen Groß- und Kleinschreibung.

Achte also beim Eintippen genau darauf, wann mal ein großer Buchstabe zwischen den vielen Kleinbuchstaben steht. Du kannst also nicht z.B. `writeline` oder `Writeline` schreiben!

Eine solche Anweisung nennt man auch Methode. In C# gibt es zahlreiche Methoden, sie gehören immer zu einem sogenannten Objekt oder zu einer Klasse – die hier `Console` heißt. Damit ist das Fenster mit dem schwarzen Hintergrund gemeint, das nur Text anzeigen kann.

Objekte kennen wir aus unserer Umgebung, z.B. Häuser, Bäume, Autos, Leute. Auch du bist ein Objekt. Und zwar vom Typ Mensch. Objekte in einer Programmiersprache sind natürlich nur künstlich.

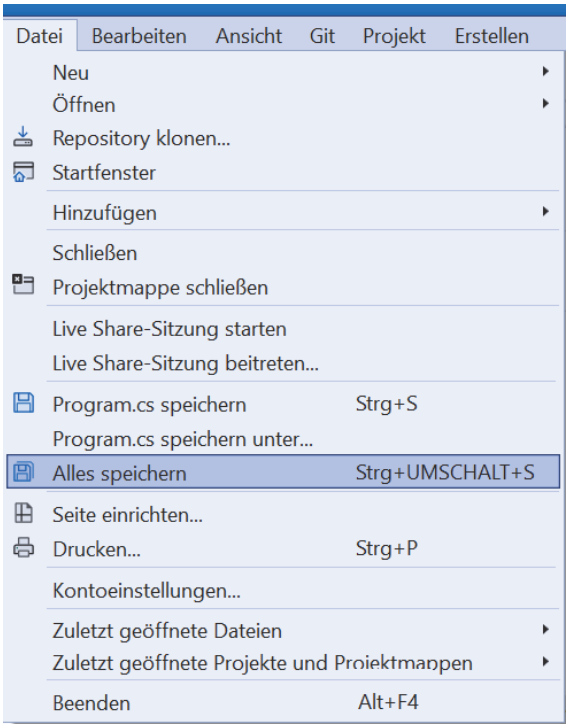
Dabei kann es in C# durchaus mehrere Objekte eines Typs geben – so wie es im richtigen Leben auch (viele) verschiedene Menschen gibt. Daher spricht man hier von **Objekttyp**, das ist dasselbe wie **Klasse**. Und ein Objekt wird auch als **Instanz** einer Klasse bezeichnet.

Demnach bist du eine Instanz der Klasse Mensch. Und mit `Console` kennst du ein Beispiel für eine C#-Klasse. Objekte lernst du später noch kennen.

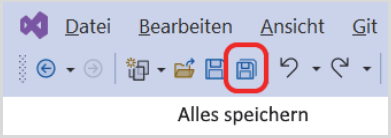
Man spricht bei `WriteLine()` auch von einer **Funktion**. Methode und Funktion meinen also hier dasselbe. Und das, was in den runden Klammern steht, wird als **Parameter** bezeichnet.

Wichtig ist: Jede (!) Methode beziehungsweise Funktion hat am Ende ihres Namens zwei runde Klammern, die können auch leer sein, sie dürfen aber niemals fehlen.

≫ Speichere dein Projekt jetzt erst mal. Dazu klickst du auf ALLES SPEICHERN im DATEI-Menü, damit werden alle beteiligten Dateien gesichert.



Alternativ dazu kannst du auch in der Symbolleiste auf den rechten der beiden Buttons mit dem Disketten-Zeichen klicken.



Disketten waren die ersten Speichermedien, das Symbol dafür hat man bis heute fürs Speichern behalten.

WRITE UND READ

Die Frage zur Begrüßung verlangt eine Antwort. Für den Namen benötigen wir eine Methode, die quasi das Gegenteil von `WriteLine()` macht. Und die heißt `Console.ReadLine()`. Es gibt also eine Anweisung für das Schreiben (auf den Bildschirm) und das Lesen (von der Tastatur).

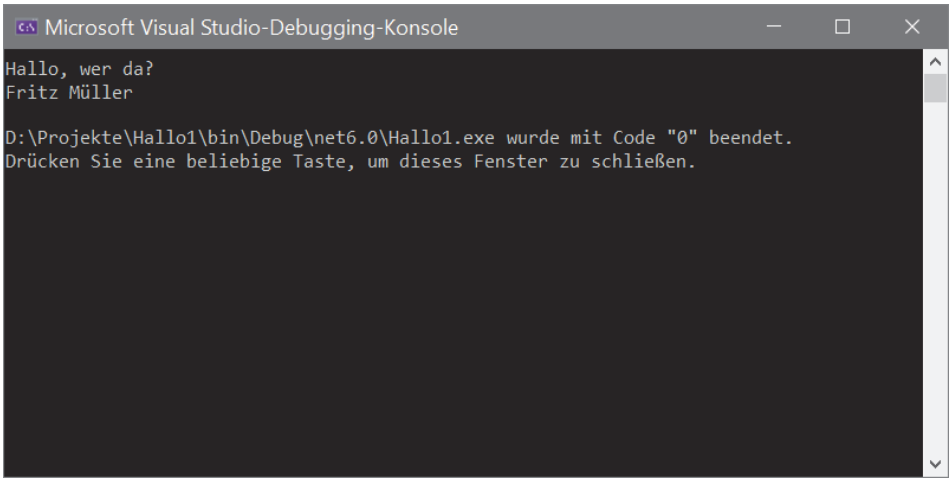
Aber genügt denn eine `ReadLine`-Anweisung allein? Probieren wir's aus. Das wäre nun unser Quelltext:

```
// Mein erstes Projekt
Console.WriteLine("Hallo, wer da?");
Console.ReadLine();
```

≫ Ändere deinen Quelltext entsprechend, speichere alles und starte dann das Programm.

Ist dir aufgefallen, wie jede der beiden Zeilen innerhalb der geschweiften Klammern beendet wird? Dieses unscheinbar aussehende Semikolon (;) schließt **jede** Anweisung ab, darf also nicht vergessen werden!

Nach einem Programmstart und der Eingabe eines Namens sieht es im Konsolenfenster so aus – wenn du die Eingabetaste gedrückt hast:



Das Drücken der Eingabetaste ist wichtig, damit der Computer weiß, dass du mit deiner Texteingabe fertig bist.

Man sieht also: `ReadLine()` funktioniert ebenso gut wie `WriteLine()`. Doch jetzt kommt ein Aber: Damit der Computer als Nächstes antworten und dabei meinen