HANS-GEORG SCHUMANN

PROGRAMMIEREN LERNEN OHNE VORKENNTNISSE

mitp

INHALT

EINLEITUNG 9

Was heißt eigentlich Programmieren? 10
Was ist eine Entwicklungsumgebung? 10
Warum gerade C#? 10
Visual Studio, die Entwicklungsumgebung zum Buch 11
Wie arbeitest du mit diesem Buch? 12
Was brauchst du für dieses Buch? 13
Hinweise für Lehrer 14

ERSTE SCHRITTE MIT C# 17 **1**

Visual Studio starten 18
Das erste Programm 20
Der Quelltext 23
Write und Read 29
Visual Studio beenden 33
Zusammenfassung 35
Ein paar Fragen 35
... aber noch keine Aufgabe 36

WENN, DANN, SONST: KONTROLLE UND AUSWAHL 37 **2**

Ein Blick zurück 38
Gut oder schlecht? 40
Die if-Struktur 43
Strings oder Zahlen? 46
Plus oder minus, mal oder durch 50
Die if-else-Struktur 52
Zusammenfassung 54
Ein paar Fragen 55
... und eine Aufgabe 55

3 ZENSUREN UND ZAHLENRATEN: BEDINGUNGEN 57
Von int zu float 58
Die Sache mit try und catch 60
Von 1 bis 6 63
Von Fall zu Fall 65
Punkt für Punkt 67
Und und Oder, oder? 69
Ein kleines Game 71
Die while-Struktur 73
Zusammenfassung 74
Ein paar Fragen 76
... und ein paar Aufgaben 76

4 GELD-SPIELEREIEN: SCHLEIFEN UND FELDER 77
Dein PC zählt mit 78
Abbruch bei Unlust? 79
Auf dem Weg zum Millionär 81
Schleifenvariationen 83
Zählen mit for 84
Variablenfelder 87
Lottoziehung 90
Feldsortierung 92
Zusammenfassung 94
Ein paar Fragen 95
... und ein paar Aufgaben 95

5 DO IT YOURSELF: FUNKTIONEN UND KLASSEN 97
C# ist lernfähig 98
Funktionen fürs Ratespiel 100
Lokal oder global? 102
Parameter und Rückgabe 104
Ein neues Baby? 106
Konstruktor 107
Eine neue Datei 110
Laufen lernen 112
Kapselung und Vererbung 114
Erbschafts-Probleme? 117
Zusammenfassung 120
Ein paar Fragen 120
... und ein paar Aufgaben 121

NICHT NUR WAS FÜRS AUGE: JETZT WIRD'S VISUELL 123 **6**
Erst mal ein Fenster 123
Von der Konsole zum Formular 125
Hallo, wie geht es? 128
Gut oder schlecht? 131
Es passiert etwas 134
Antwort nach Maß 138
Zusammenfassung 139
Ein paar Fragen 140
... und eine Aufgabe 140

ALLES UNTER KONTROLLE: KOMPONENTENSAMMLUNG 141 **7**
Kleine Knopfparade 141
Alles in einer Liste 144
Du hast die Wahl 147
Optionen 150
Von Pünktchen und Häkchen 154
Körper, Geist und Seele 156
Der letzte Schliff 159
Zusammenfassung 160
Ein paar Fragen 161
... aber nur eine Aufgabe 161

WER WEISS WAS? QUIZ-PROJEKT TEIL 1 ..... 163 **8**
Erst der Plan, dann der Bau 163
Frage und Antworten 166
Richtig oder falsch 168
Die passende Textdatei 170
Datensammlung 173
Datentransfer 176
Aufsammeln und einordnen 178
Zusammenfassung 181
Ein paar Fragen 182
... aber keine Aufgaben 182

SPIELEN UND LERNEN: QUIZ-PROJEKT TEIL 2 ..... 183 **9**
Eine oder viele? 183
Lösungs-Zähler 186
Aufgabenkontrolle 189
Antwort als Optionen 192

Vokabeln lernen? 194
Mehrfachauswahl 198
Zusammenfassung 203
Ein paar Fragen 203
... aber keine Aufgaben 203

10 JETZT WIRD'S BUNT: GRAFIK IN C# 205
Von Punkten und Koordinaten 206
Das erste Bild 208
Linienführung 211
Jetzt wird's bunt 212
Eckig und rund 214
Mit Text geht auch 216
Farbtupfer 218
Selbst zeichnen? 220
Zusammenfassung 224
Ein paar Fragen 225
... und ein paar Aufgaben 225

11 BILDER LERNEN LAUFEN: ANIMATIONEN 227
Erst mal ein Kreis 227
Und es bewegt sich was 230
PictureBox 232
Endlich ein (richtiges) Bild 234
Bildersammlung 237
Da läuft was 239
Drehen, verschwinden, auftauchen 241
Movie komplett 243
Zusammenfassung 245
Keine Fragen 245
... doch ein paar Aufgaben 245

12 BUNTES DUO: EIN PAAR SPIELE 247
Komponenten zusammenbauen 247
Wie viele Augen? 250
Anzeigen und auswerten 254
Schere – Stein – Papier 257
Qual der Wahl 260
Komponenten-Arrays 262
Zusammenfassung 265
Keine Fragen 266

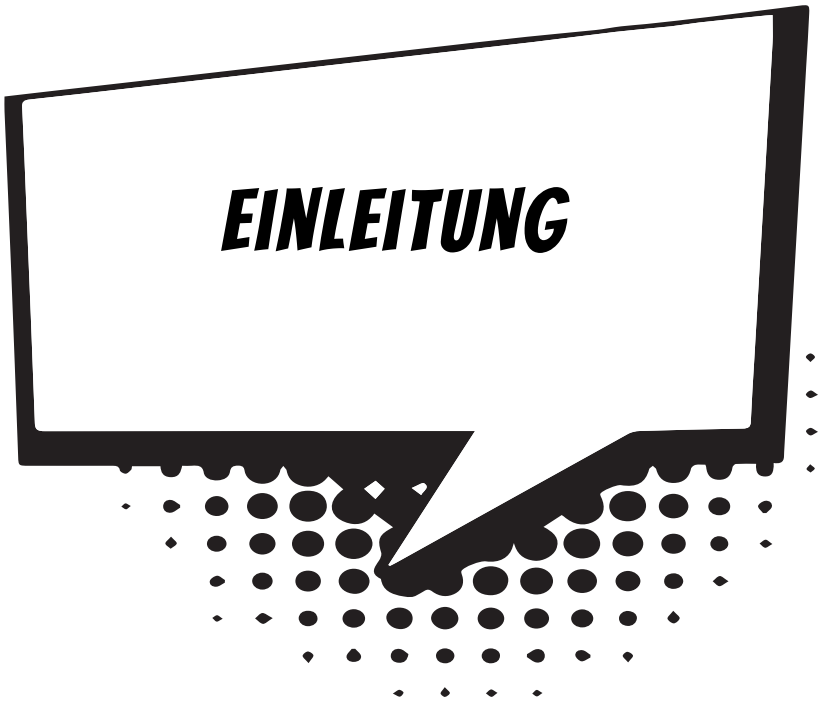
... aber zwei Aufgaben 266

KLEINER KRABBELKURS: TASTEN- UND MAUSSTEUERUNG 267 13
Eine Spinne am Start 268
Tastensteuerung 270
Richtungswechsel 273
Kein Spiel ohne Grenzen 276
Maussteuerung 278
Die Sache mit dem Timer 281
Zusammenfassung 285
Keine Fragen 286
... und keine Aufgaben 286

ANHANG A ..... 287 A
Visual Studio installieren 287
Einsatz der Buch-Dateien 292

ANHANG B ..... 293 B
Kleine Checkliste 293
Dem Fehler auf der Spur 294

STICHWORTVERZEICHNIS 299



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

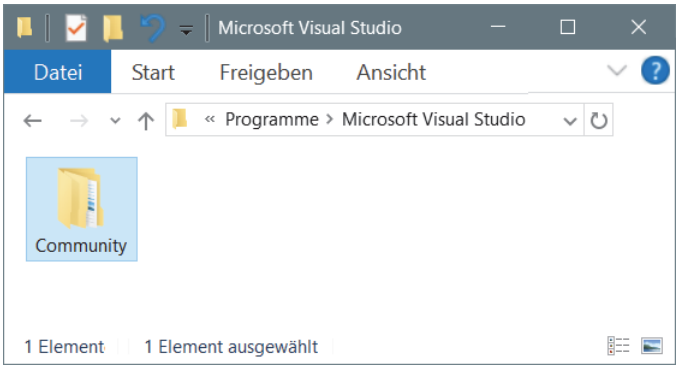
- wie man Visual Studio startet
- wie man ein Programm erstellt und ausführt
- einiges über (mögliche) Fehler
- was `WriteLine()` und `ReadLine()` bedeuten
- etwas über Variablen
- wie man ein Projekt speichert
- wie man Visual Studio beendet

VISUAL STUDIO STARTEN

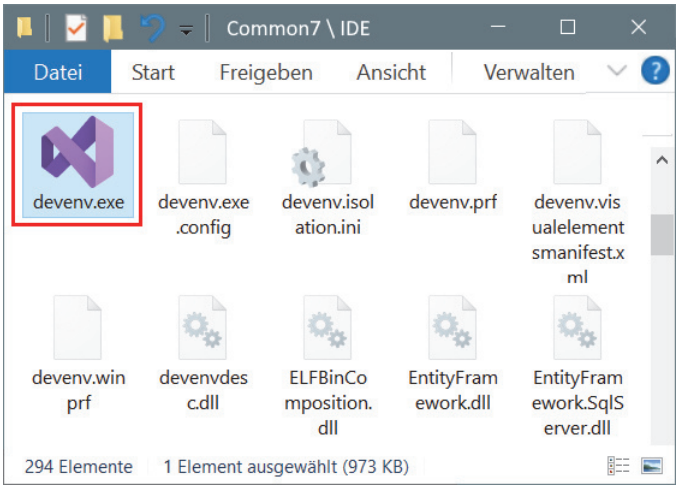
Bevor wir mit dem Programmieren anfangen können, muss **Visual Studio** erst installiert werden. Die Installation übernimmt ein Programm namens SETUP. Genaues erfährst du im Anhang A. Hier musst du dir von jemandem helfen lassen, wenn du dir die Installation nicht allein zutraust.

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).

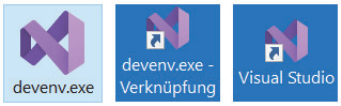


≫ Dort musst du nun weiter über einige Unterordner wie COMMUNITY und COMMON7 in IDE wechseln:



≫ 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:

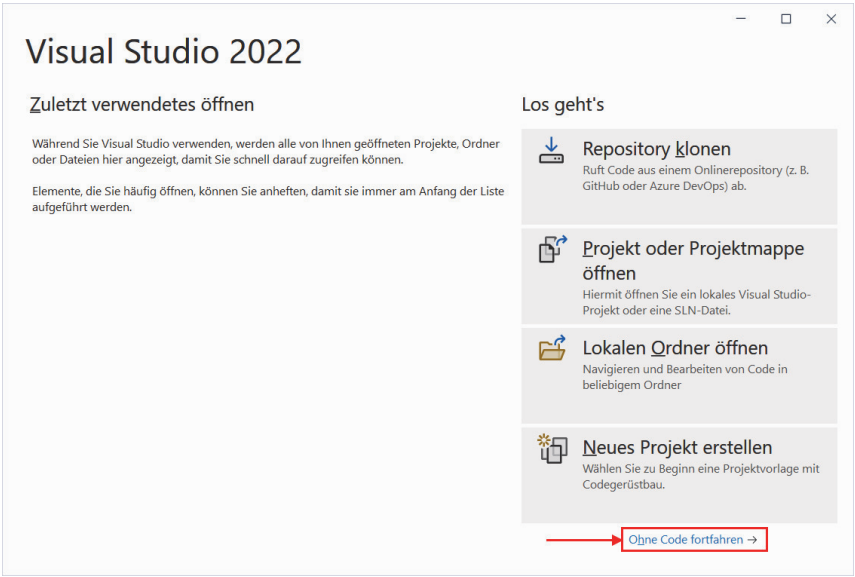


Ich empfehle dir, eine **Verknüpfung** auf dem Desktop anzulegen:

- 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:

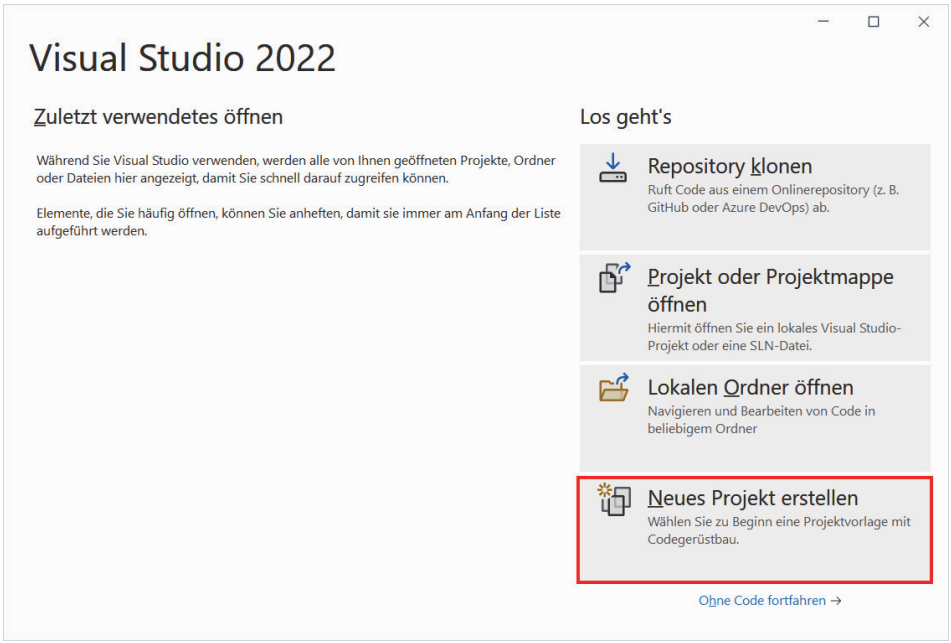
- Über das DATEI-Menü kannst du Dateien speichern, laden (öffnen), ausdrucken, neu erstellen oder Visual Studio beenden.
- Das BEARBEITEN-Menü hilft dir bei der Bearbeitung deines Programmtextes, aber auch bei anderen Programmelementen. Außerdem kannst du dort bestimmte Arbeitsschritte rückgängig machen 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!

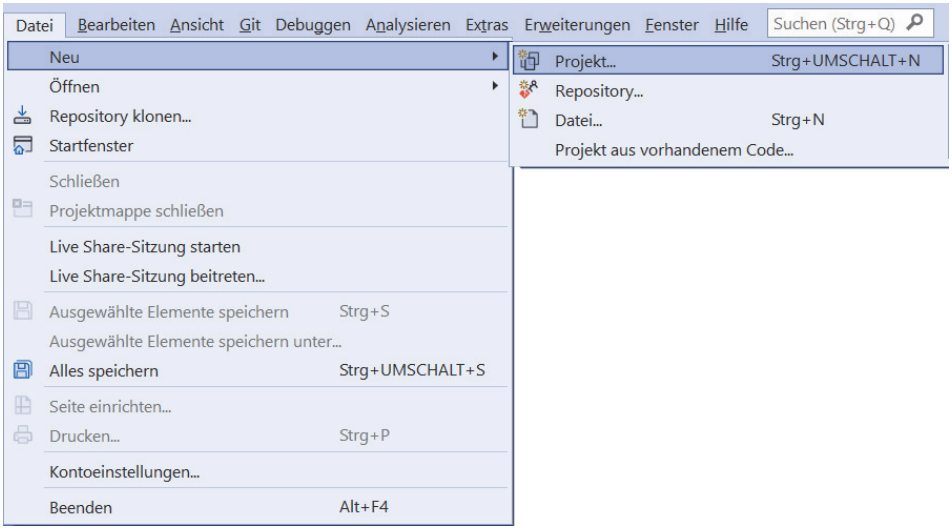
DAS ERSTE PROGRAMM

≫ 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:

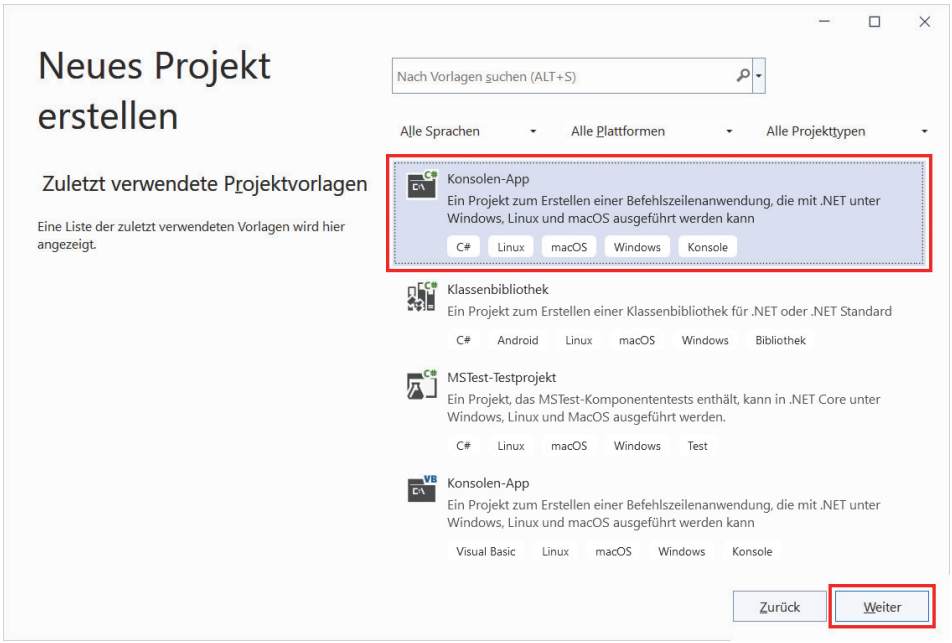


≫ Dann klickst du auf NEUES PROJEKT ERSTELLEN.

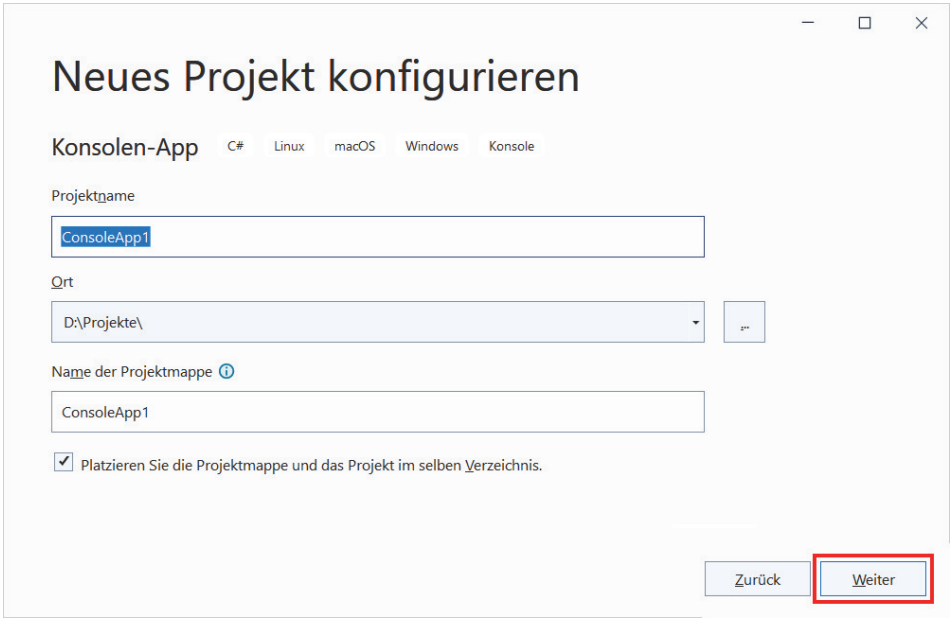
≫ Oder du hast dich entschieden, bei der Menüleiste von Visual Studio zu bleiben. Dann klickst du dort auf DATEI und im sich öffnenden Menü auf NEU und dann auf PROJEKT.



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.



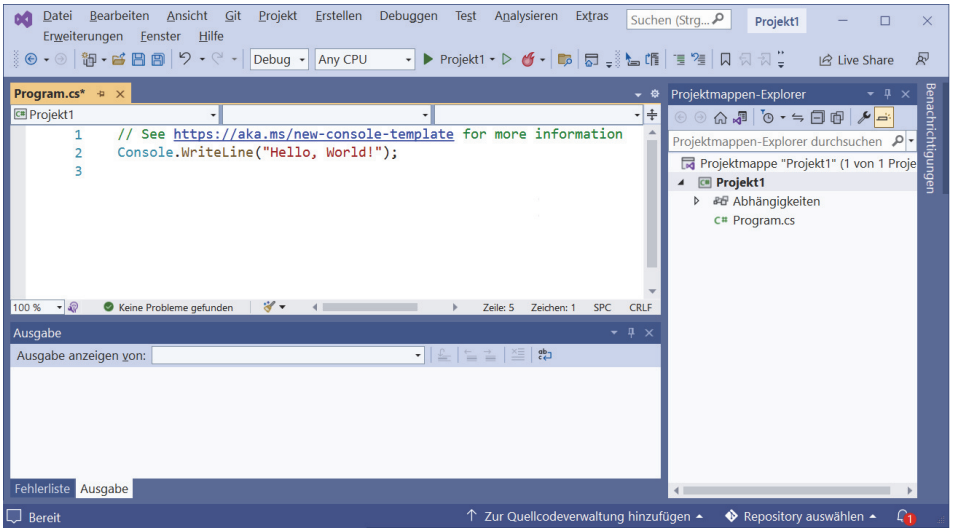
≫ Gib dem Projekt einen Namen und sorge dafür, dass hinter SPEICHERORT der Pfad steht, wo du dein Projekt unterbringen willst. Dann klicke abschließend auf WEITER.

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:.

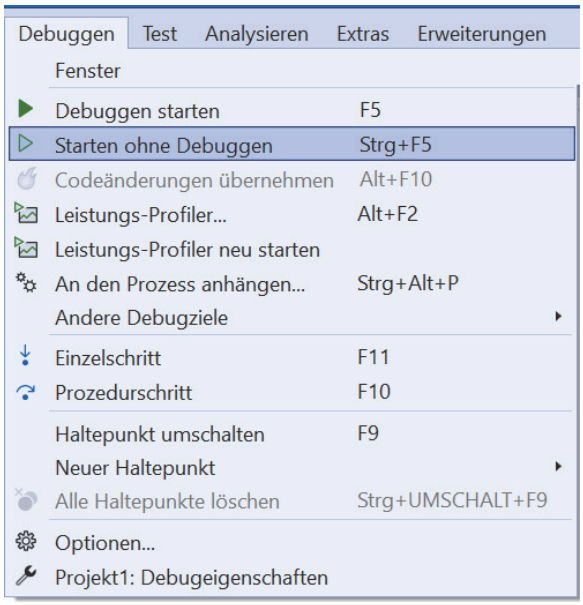
≫ Im nächsten Dialogfeld erzeugst du mit einem abschließenden Klick auf ERSTELLEN dein erstes Mini-Projekt in C#.

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
```

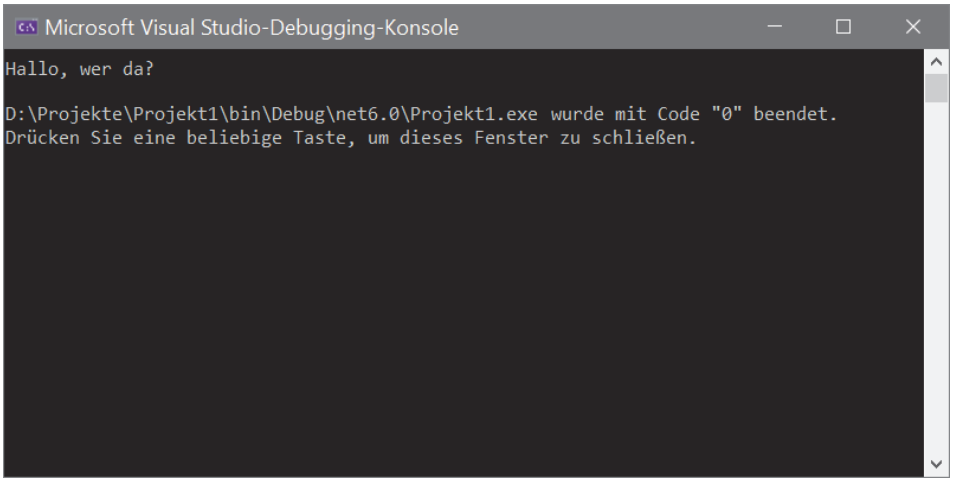
Oder man lässt diesen Kommentar einfach ganz weg. Für ein funktionierendes Programm ist nur diese Zeile nötig:

```
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:

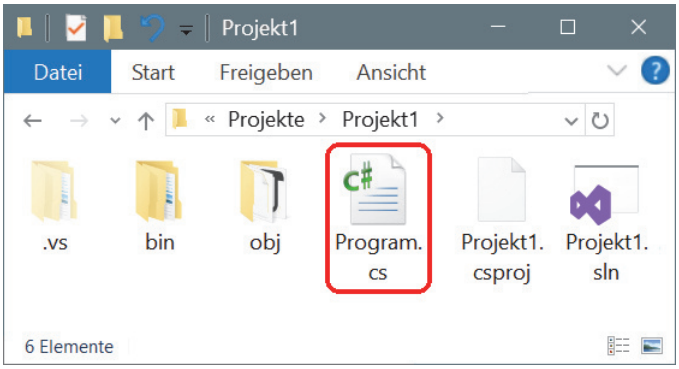
```
Console.WriteLine("Hallo, wer da?");
```

≫ 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.

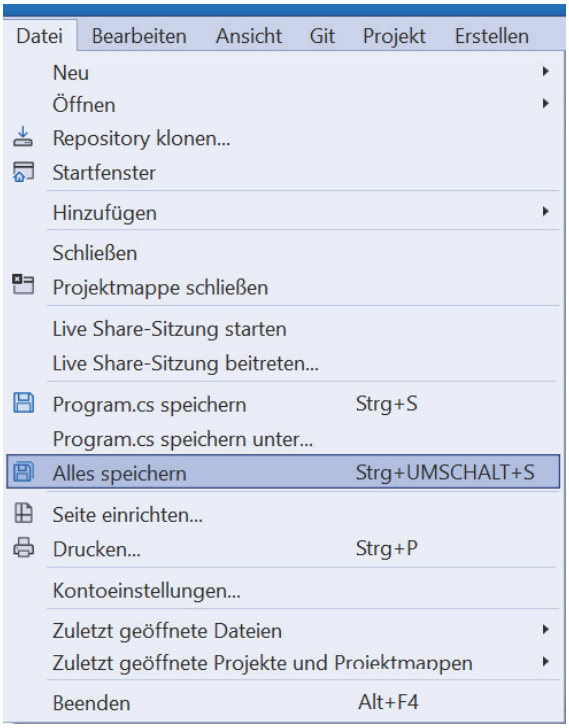
Dabei kann es in C# durchaus mehrere Objekte eines Typs geben – so wie es im richtigen Leben auch (viele) verschiedene Menschen gibt. Daher spricht man hier von **Objekttyp**, das ist dasselbe wie **Klasse**. Und ein Objekt wird auch als **Instanz** einer Klasse bezeichnet.

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.

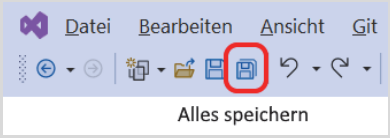
Man spricht bei `WriteLine()` auch von einer **Funktion**. Methode und Funktion meinen also hier dasselbe. Und das, was in den runden Klammern steht, wird als **Parameter** bezeichnet.

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.

STICHWORTVERZEICHNIS

@ 176
$ 48

A

Abbruch 79, 80
Add 175, 251
Addition 51
AddLine 223
and 69
Anweisung 27
Anweisungsblock 45, 74
Anzeigefeld 164
Argument 104
Array 89, 148, 167, 273
Array.Sort 93
Attribut 107
Auflösung 206
AutoSize 186

B

BackColor 218
base 115
Bedingung 45, 74
Bildfeld 232
Bildpunkt 206
Bit 170
Bitmap 237
Black 212
bool 44, 154, 158, 189
BorderStyle 186, 249
break 66, 67, 79
Bug 294
Button
    Font 133
    Text 133
button_Click 137

C

C# 9, 11
    Aussprache 26
case 65
catch 62
CheckBox 155
Checked 153, 194
CheckedChanged 153, 157, 193
class 106
Click 256
ClientSize 212, 231
Clone 274
Close 178
Codierung 172
CollectImages 238, 252
Color 212, 213
ComboBox 149
Community 287
Compiler 10
Console 27
const 90, 213
continue 80
Controls 251
Controls.Add 251
Convert 170
Count 179
CreateGraphics 229

D

Debug
    beenden 295
    Einzelschritt 295
    Haltepunkt 296
    Prozedurschritt 294
Debuggen 41
Debugger 294
default 67
Dekrementieren 79
Delay 237

Destruktor 108
Dimension 167
Division 51
Doppelklick
    Ereignis-Verbindung 137
Doppelpunkt 66, 115
DoubleBuffered 272
do-while 82
DrawEllipse 215
DrawImage 269
DrawLine 211
DrawPath 221
DrawRectangle 215
DrawString 217

E

e.X 279
e.X, e.Y 222
e.Y 279
Editor 10
Eigenschaft 107
Eigenschaften 129, 132, 143
Einzelschritt (Debug) 295
else 53
else if 255
Endlos-Schleife 80
Entwicklungsumgebung 10
Ereignis 134, 256
    Verbindung 137
Ereignis-Methode 136
Evaluate 187, 254
Event 134, 256
EventHandler 138, 256, 257
Exception 49, 60, 61

F

Fallunterscheidung 65
false 154
Fehler 30, 32, 44
Fehler abfangen 61
Fehlertyp 62
Feld 89
Feldvariable 89
File 178
FillEllipse 219
FillRectangle 219
float 58
float.Parse 58
Font 217, 249
for 85
foreach 175
Form1 namespace 196
Form_Load 136
Form_Paint 209, 269
Formular 125
    Eigenschaften 129
    Titel 129
for-Struktur 86
FromFile 239
Funktion
    Aufruf 100
    Definition 99
Funktionskopf 99
Funktionsrumpf 100

G

Globale Variable 102
Grafik
    Ellipse 211
    Farbe 212
    Linie 211
    Rechteck 211
Grafikkarte 206
Graphics 209, 224, 268
    DrawEllipse 211
    DrawLine 211
    DrawRectangle 211
    DrawString 217
GraphicsPath 222
Groß-klein 27
Grundrechenart 51

H

Haltepunkt (Debug) 296
Height 212
HideImage 231
Hilfe 294

I

if 43
if-Struktur 46
Image 236, 237

in 175
Index 89, 90
InitGame 100
Initialisierung 90
    Startwerte 100
Inkrementieren 79
Installation 287
Instanz 27, 114
int 47
int.Parse 47
Interval 283
Invalidate 223

J

JPG 235

K

Kapselung 114
KeyCode 271
KeyDown 270
KeyEvent 271
KeyPress 270
Keys 271
KeyUp 270
Klammern
    geschweifte {} 39, 45, 61, 66
    runde () 28, 45, 99
    spitze 174
Klasse 27, 114
Knoten 131
Kommentar 25
Komponente 124
    Eigenschaften 132
    Font 133
    Text 133
Konsolenfenster 24, 46
Konstante 90, 213
Konstruktor 107, 115, 166
Kontrollfeld 155
Kontrollstruktur 46, 65, 74, 82, 175
Koordinaten 206
Kopieren 59

L

Label 164
Leerkette 31
Leerzeichen 50
Length 91, 179
Lines 177
List 174
ListBox 145
Listenfeld 145
Load 235
Location 143, 235, 249
Lokale Variable 102

M

Main 39, 127
Markierungsfeld 156
Maussteuerung 278
Mehrfachauswahl 195
MessageBox 138
Methode 27, 107
    Verknüpfung lösen 202
Modul 98
Mouse
    Position 222
MouseClick 279
MouseDown 220
MouseEvent 278
MouseEventArgs 223
MouseMove 220
MouseUp 220
MoveImage 231, 280
Multiplikation 51

N

namespace 127
new 72, 89, 109, 158, 174
Next 71
Not-Operator 190
null 236

O

Objekt 27, 109
Objektorientierte Programmierung 114
Objekttyp 27
Oder-Operator 70
Operator
    -- 79
    ! 190
    != 53, 54

. 47
&& 70
++ 79
+= 79, 256, 271
<> 54
-= 79, 271
= 33, 44
== 54
> 54
|| 70
Rechnen 51
Vergleich 54
Zuweisung 33
Optionsfeld 150, 192
Optionsgruppe 152
or 69
override 119

P

PaintEventArgs 209
Parameter 28, 104
Parse 47
Pen 212
PictureBox 232
Load 235
Pixel 206
PlayGame 100
PNG 235
Point 235
Popup 20
private 136
Programm
Abbruch 63
starten 24
Programmieren 10
Programmiersprache 10
Programmierung
objektorientiert 114
Projekt
Kopie 59
neu 21
öffnen 39
speichern 28, 34
starten 24
Projekt-Kopie 195
protected 136
Prozedur 105, 295
Prozedurschritt (Debug) 294
public 117, 136
Punkt 235
Punktobjekt 235

Q

Quelltext 25
automatisch 251
Quiz 163

R

RadioButton 150
Random 71, 87, 184
ReadAllLines 177
ReadLIne 29
ReadLine 177
Refresh 240, 272
return 105, 190
RotateFlip 273
RotateFlipType 273
Rückgabewert 105, 117
Run 127

S

Schaltfeld 156
Schalt-Variable 189, 222
Schere-Stein-Papier 257
Schleife 82
Schlüsselwörter 98
SelectedIndex 148, 149
SelectedIndexChanged 148
Semikolon 29
sender 265
SetLimits 276
Setup 18
Show 138
ShowImage 229, 234
Size 143, 249
SizeMode 233, 250
Sleep 236, 281
SLN 39
SolidBrush 217
Sort 92
Sortieren 92
Spickzettel 109
Start 283

Steuerelement 124
Steuerung
    Maus 278
Stop 284
StreamReader 177
StreamWriter 177
StretchImage 250
String 31
Subtraktion 51
switch 65, 149
switch-Struktur 65
System.IO 178

T

TabStop 154
Task 237
Terminate 63
Text anzeigen 217
TextAlign 165, 249
TextBox 177
Textdatei 171
then 43
this 139
Thread 236
Tilde 108
Timer 281
timer_Tick 282
Toolbox 130
ToString 170
Transparenz 235
true 154
try 61
try-catch 234
TurnImage 241
Typumwandlung 170

U

Und-Operator 70
Unicode 172
Ursprung 206
using 127, 236, 281

V

Variable 31
    global 102
    lokal 102
Variablenfeld 89
Vereinbarung 31
Vergleichsoperator 44, 54
Verknüpfung 19, 210
    Event 256
Verknüpfungsoperator 47, 70
Verzweigung 65
virtual 119
Visual Studio 11
    beenden 33
    Fenstergruppe 23
    installieren 287
    Menüs 20
    neues Projekt 21
    Projektstart 24
    starten 18
    Verknüpfung 19
void 100

W

Wahlfeld 156
Warnung 32
Wellenlinien
    grüne 32
    rote 32
while 73
while-Struktur 74
Width 212
Windows Forms 125, 141
Wortschatz 98
WPF 125
Write 50
WriteAllLines 177
WriteLine 27, 177
Würfelspiel 247

X

x-Achse 206

Y

y-Achse 207

Z

z-Achse 207
Zählschleife 86

Zugriff
    öffentlich 136
    privat 136
Zugriffsmodus 136
Zugriffsoperator 47
Zuweisung 32, 44, 51, 78