C++ Programmieren für Einsteiger

Michael Bonacina

2. Auflage: August 2018

© dieser Ausgabe 2019 by BMU Media GmbH

ISBN: 978-3-96645-005-8

Herausgegeben durch:
BMU Media GmbH
Hornissenweg 4
84034 Landshut

C++ Programmieren für Einsteiger

Inhaltsverzeichnis

1. Einleitung 9

1.1 Eine der am häufigsten verwendeten Programmiersprachen: C++ 9
1.2 Die Geschichte von C++ 10
1.3 Was zeichnet C++ aus? 11
1.4 Programmieren lernen mit C++ 13

2. Die Vorbereitung: Diese Programme sind für das Programmieren in C++ nötig 15

2.1 Ein Texteditor für die Gestaltung des Programmcodes 15
2.2 Ein Compiler für die Erstellung der Programme 17
2.3 Visual Studio: eine integrierte Entwicklungsumgebung 23

3. Das erste eigene Programm in C++ schreiben 26

3.1 Der erste Schritt: eine einfache Ausgabe auf dem Bildschirm 26
3.2 Die Elemente des Programmcodes verstehen 27
3.3 Das Programm ausführen 29
3.4 Übungsaufgabe: So lässt sich der Programmcode verändern 31

4. Die Verwendung von Variablen in C++ 35

4.1 Wozu dienen Variablen? 35
4.2 Variablen: verschiedene Typen 36
4.3 Variablen in das Programm integrieren 38
4.4 Operatoren für die Bearbeitung von Variablen 42
4.5 Arrays: Blöcke aus mehreren Variablen 47
4.6 Übungsaufgabe: Programme mit Variablen erstellen 51

5. Entweder oder: Entscheidungen im Programm treffen 57

5.1 Verzweigungen mit einer if-Abfrage erstellen 57
5.2 Vergleiche für das Aufstellen einer Bedingung 58
5.3 Mehrere Bedingungen miteinander verknüpfen 62
5.4 Alternativen für die Ausführung mit else einfügen 65
5.5 Übungsaufgabe: Abfragen und Verzweigungen verwenden 69

6. Befehle wiederholen: mit Schleifen arbeiten 75

6.1 While-Schleifen: die Grundform aller Schleifen 75
6.2 Do-while: Bedingungen am Ende der Schleife vorgeben 78
6.3 For-Schleifen für eine feste Anzahl an Durchläufen 81
6.4 For-each: spezielle Schleifen für die Arbeit mit Arrays 83
6.5 Übungsaufgabe: Schleifen im Programm anwenden 86

7. Funktionen in C++ verwenden 93

7.1 Was ist eine Funktion und welche Vorteile bietet sie? 93
7.2 Funktionen selbst erstellen 94
7.3 Funktionen in externen Dateien abspeichern 99
7.4 Vorgefertigte Funktionen aus Bibliotheken nutzen 100
7.5 Übungsaufgabe: eigene Funktionen erstellen 103

8. Strukturen: eigene Datentypen in C++ gestalten 108

8.1 Datensätze aus verschiedenen Typen 108
8.2 Strukturen als Vorlage erstellen 109
8.3 Die Verwendung von Strukturen im Programm 110
8.4 Übungsaufgaben: Datensätze durch Strukturen vereinfachen 111

9. Objektorientierung in C++ 116

9.1 Was bedeutet Objektorientierung? 116
9.2 Die Klasse: Vorlage für Objekte 118
9.3 Die Attribute eines Objekts 119

9.4 Methoden: spezielle Funktionen für Objekte 123
9.5 Vererbung von Attributen und Methoden 129
9.6 Übungsaufgabe: mit Objekten arbeiten 134

10. Zeiger verwenden 143

10.1 Was sind Zeiger und wozu dienen sie? 143
10.2 Adressen von Variablen 144
10.3 Zeiger in Funktionen verwenden 149
10.4 Speicherplatz dynamisch vergeben 153
10.5 Übungsaufgabe: mit Zeigern arbeiten 155

11. Daten dauerhaft abspeichern: die Verwendung von Dateien 162

11.1 Daten speichern 162
11.2 Daten einlesen 166
11.3 Übung: Dateien in Programme einbinden 168

12. Visual Studio: eine IDE für die Programmierung in C++ 174

12.1 Welche Vorteile bietet Visual Studio? 174
12.2 Ein Projekt mit Visual Studio erstellen 175
12.3 Programme kompilieren und ausführen 179

13. Grafische Benutzeroberflächen mit MFC erstellen 183

13.1 GUIs mit C++ erstellen: verschiedene Möglichkeiten 183
13.2 Eine einfache grafische Benutzeroberfläche erstellen 185
13.3 Das Fenster individuell gestalten und eigene Elemente hinzufügen 189
13.4 Auf Ereignisse reagieren 194
13.5 Übungsaufgabe: Ein eigenes Programm mit Fenstern erstellen 200

14. Anwendungsbeispiel: ein Programm für die Lagerverwaltung 204

14.1 Das grundlegende Fenster gestalten 205
14.2 Bestand anzeigen lassen 206
14.3 Einen neuen Artikel ins Sortiment aufnehmen 217
14.4 Bestand auffüllen und Artikel verkaufen 225

15. Ausblick: Wie geht es weiter? 240

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

http://bmu-verlag.de/cpp_programmieren/



Kapitel 1
Einleitung

C++ ist eine Programmiersprache, die Softwareentwicklern vielfältige Möglichkeiten bietet. Im Gegensatz zu anderen Programmiersprachen, die auf ein ganz bestimmtes Anwendungsgebiet spezialisiert sind, eignet sich C++ für beinahe alle Bereiche. C++ ist ein sehr mächtiges Werkzeug und ermöglicht es, ausgesprochen effiziente Programme zu entwickeln. Dieses Buch bietet eine Einführung in die Programmierung in C++. Es richtet sich dabei vorwiegend an Anfänger. Programmierkenntnisse werden dabei nicht vorausgesetzt. Lediglich der grundlegende Umgang mit den Funktionen eines Computers sollte bekannt sein. Es stellt einen einfachen Einstieg in die Programmierung dar und bietet die Möglichkeit, eine häufig verwendete Programmiersprache mit vielfältigen Anwendungsmöglichkeiten zu erlernen.

1.1 Eine der am häufigsten verwendeten Programmiersprachen: C++

Wie bereits im ersten Abschnitt erwähnt, eignet sich C++ für sehr vielfältige Anwendungen. Die Programmiersprache erlaubt zum einen die Erstellung von Anwendungsprogrammen. Dazu zählt ganz unterschiedliche Software – von Programmen für die Büroverwaltung oder für die Organisation eines Warenlagers bis hin zu Computerspielen. Etwa bis zur Jahrtausendwende dominierte C++ diesen Bereich klar. Seitdem konnten andere Programmiersprachen wie Java oder C# stark zulegen. Allerdings kommt C++ hierfür auch weiterhin häufig zum Einsatz.

Die Systemprogrammierung umfasst Programme, die entweder direkt auf Hardware- oder auf Betriebssystemfunktionen zugreifen. Hierzu zählt die Entwicklung der Betriebssysteme selbst, die Gestaltung von Treibern sowie die Erstellung von virtuellen Maschinen, eingebetteten

Systemen und ähnlichen Programmen. Diese Art von Software stellt ganz andere Anforderungen an die Programmiersprache. Doch auch in diesem Bereich stellt C++ die notwendigen Funktionen bereit. Daher kommt diese Sprache auch hierbei sehr häufig zum Einsatz und nimmt dabei eine führende Rolle ein.

Die vielfältigen Anwendungsmöglichkeiten führen dazu, dass C++ eine ausgesprochen beliebte Programmiersprache ist und für sehr viele Programme zum Einsatz kommt. C++ zu erlernen, ist daher sehr sinnvoll – unabhängig davon, ob man Anwenderprogramme auf einem höheren Abstraktionsniveau programmieren oder auf die grundlegenden Funktionen des Computers zugreifen will. Wer die Grundlagen von C++ beherrscht, kann sich später auf einen dieser Bereiche spezialisieren.

1.2 Die Geschichte von C++

Die Entwicklung von C++ geht auf den dänischen Informatiker Bjarne Stroustrup zurück. Dieser war im Rahmen seiner Doktorarbeit an der Cambridge University auf das Problem gestoßen, dass er für die von ihm angestrebten Projekte keine geeignete Programmiersprache fand. Dabei handelte es sich um Software-Projekte, die zum einen sehr umfangreich waren und bei denen zum anderen viel Wert auf eine hohe Effizienz gelegt wurde. Stroustrup verwendete zum einen die Programmiersprache Simula, die zwar gut für umfangreiche Programme geeignet war, aber bei der es nur sehr schwer möglich war, eine hohe Effizienz zu erreichen. Zum anderen kam die Sprache BCPL zum Einsatz, bei der zwar eine hohe Effizienz möglich war, bei der sich die Verwirklichung großer Projekte jedoch als schwierig erwies.

Aufgrund dieses Problems beschloss er 1979 im Rahmen seiner Tätigkeit bei AT&T Bell Laboratories, eine neue Programmiersprache zu entwickeln. Diese sollte beide erwähnten Eigenschaften miteinander verbinden. Allerdings entschied er sich dazu, diese nicht von Grund auf neu zu entwerfen. Anstatt dessen entwickelte er die Programmiersprache C weiter. Bis heute ist C++ mit C kompatibel, sodass es möglich ist, C-Code in ein C++-Programm einzubauen. Die Wahl fiel aus verschie-

denen Gründen auf C: Zum einen lag diese Sprache dem damals häufig genutzten Betriebssystem UNIX bei und war daher weit verbreitet. Zum anderen war es relativ einfach, C-Programme auf weitere Betriebssysteme zu portieren. Schließlich eignet sich C für viele unterschiedliche Anwendungen und ermöglicht es, Programme mit hoher Effizienz zu erstellen.

Stroustrup griff dabei ein Konzept auf, das erst wenige Jahre zuvor entwickelt wurde: die Objektorientierung. Was das genau ist, wird später noch etwas ausführlicher behandelt. An dieser Stelle sei lediglich erwähnt, dass es sich hierbei um ein Programmierparadigma handelt, das zu Beginn der 70er Jahre entstand und das für moderne Programmiersprachen unverzichtbar ist. Es beruht im Wesentlichen auf Klassen, Objekten und Methoden. Die Objektorientierung war so prägend für die Entwicklung von C++, dass Stroustrup die Sprache zunächst "C with Classes" – also C mit Klassen – taufte. 1983 erfolgte dann die Umbenennung zu C++. 1990 erschien das Buch The Annotated C++ Reference Manual, das die wichtigsten Grundlagen der Programmiersprache vorgab. Die Standardisierung von C++ erfolgte erst 1998 – beinahe 20 Jahre nachdem Stroustrup mit der Entwicklung der Programmiersprache begonnen hatte.

1.3 Was zeichnet C++ aus?

Wie bereits zu Beginn der Einführung erwähnt, eignet sich C++ für die Gestaltung sehr effizienter Programme. Um zu verstehen, weshalb dies so ist, ist es notwendig, kurz auf die grundlegende Funktionsweise eines Computerprogramms einzugehen. Ein Programm in C++ oder in einer anderen Programmiersprache besteht im Grunde lediglich aus Text. Dieser muss zwar nach ganz speziellen Strukturen aufgebaut sein, doch wird das Programm als Text abgespeichert und die Bedeutung der einzelnen Schlüsselbegriffe ist für Personen mit entsprechenden Programmierkenntnissen einfach zu verstehen. Der Computer versteht – auf Ebene der Hardware – diese Kombinationen aus Zahlen und Buchstaben jedoch nicht. Hierbei ist ein ganz spezifischer Code aus binären Informationen notwendig, der auch als Maschinensprache bezeichnet

wird. Dieser enthält ganz spezifische Anweisungen – beispielsweise wo bestimmte Daten abgerufen werden sollen, auf welche Weise sie bearbeitet werden sollen und wo das Ergebnis abgelegt werden soll.

Um ein Computerprogramm auszuführen, ist es notwendig, den Text mit den Anweisungen in Maschinensprache zu übersetzen. Hierfür gibt es verschiedene Möglichkeiten. Bei sogenannten Interpreter-Sprachen kommt ein Programm zum Einsatz, das den Code einliest und während der Anwendung übersetzt. Das bringt den Vorteil mit sich, dass diese Programme auf jedem Computer laufen, auf dem ein entsprechendes Übersetzungsprogramm installiert ist – unabhängig vom Betriebssystem. Eine andere Möglichkeit besteht darin, das Programm zu kompilieren – was beispielsweise bei C++ der Fall ist. Das bedeutet, dass es bereits zu Beginn in Maschinencode übersetzt wird. Dabei entsteht ein ausführbares Programm, das spezifische Anweisungen für das jeweilige Betriebssystem enthält. Darüber hinaus gibt es auch Zwischenlösungen wie bei der Programmiersprache Java, die das Programm vorkompilieren und anschließend auf einer virtuellen Maschine ausführen.

Diese Funktionsweise mag für Anfänger im Bereich der Informatik schwer zu verstehen sein. Allerdings ist diese Ausführung wichtig, um zu verstehen, weshalb C++ sehr effizient ist. Hierbei wird ein fertig übersetztes Programm erzeugt, das bereits in Maschinencode vorliegt. Bei Interpreter-Sprachen ist es hingegen bei jeder Ausführung erneut notwendig, den Code zu übersetzen. Das nimmt viel Zeit in Anspruch und beeinträchtigt daher die Effizienz. Darüber hinaus erlaubt C++ einen direkten Zugriff auf den Prozessor. Bereits diese Eigenschaft für sich genommen erhöht die Ausführungsgeschwindigkeit. Außerdem ist es so möglich, den Ablauf des Programms sehr genau zu steuern und so zu optimieren, dass die Ausführung sehr schnell ist und so wenig Ressourcen wie möglich in Anspruch nimmt. Daher eignet sich diese Programmiersprache hervorragend für Anwendungen, bei denen eine hohe Effizienz notwendig ist.

Ein weiterer Vorteil von C++ besteht in der hohen Flexibilität. Die Sprache unterstützt neben der Objektorientierung auch einige weitere Pro-

grammierparadigmen. Daher hat er Programmierer relativ freie Auswahl und kann die Konzepte umsetzen, die ihm persönlich zusagen und die sich für die jeweiligen Aufgaben eignen.

1.4 Programmieren lernen mit C++

Programmieren zu lernen, bringt ohne jeden Zweifel viele Vorteile mit sich – sowohl im Privatleben als auch für die berufliche Karriere. Wer diesen Entschluss fasst, muss sich jedoch für eine Programmiersprache entscheiden. Dabei gibt es unzählige Möglichkeiten. Das macht die Entscheidungsfindung nicht einfach.

Wer dieses Buch in der Hand hält, wird die Programmiersprache C++ verwenden, um die grundlegenden Techniken der Programmierung zu erlernen. Das stellt aus mehreren Gründen eine gute Wahl dar.

Ein wichtiger Aspekt ist dabei die Verbreitung. C++ zählt zu den verbreitetsten Programmiersprachen und kommt für unzählige Anwendungen zum Einsatz. Wer diese Sprache erlernt und dabei ein hohes Niveau erreicht, findet daher sicherlich viele Beschäftigungsmöglichkeiten. C++-Programmierer sind sehr gefragt, sodass die entsprechenden Kenntnisse die Karrierechancen deutlich verbessern.

Ein weiterer Punkt, der für C++ spricht, ist die hohe Flexibilität dieser Sprache. Daher ist es zu Beginn nicht notwendig, sich für einen bestimmten Bereich zu entscheiden. Vielmehr ist es sinnvoll, die Funktionen und Eigenschaften der Programmiersprache zunächst intensiv kennenzulernen. Erst danach ist es erforderlich, sich zu entscheiden, auf welche Anwendungen man sich spezialisieren will.

Außerdem ist es vorteilhaft, dass es sich hierbei um eine Compilersprache handelt. Syntaxfehler, die das Programm eventuell enthält, werden daher bereits beim Kompilieren erkannt und können berichtigt werden. Das erleichtert die Fehlersuche und führt dazu, dass es Anfängern leichter fällt, korrekte Programme zu erstellen.

Dass C++ zur C-Sprachfamilie gehört, ist ebenfalls sehr hilfreich. Auf C bauen sehr viele weitere Sprachen auf – neben C++ auch Java, C#, PHP, Perl und einige weitere. Diese weisen große Ähnlichkeiten auf. Wer zu Beginn eine Sprache aus diesem Bereich wählt, kann später mit vergleichsweise geringem Aufwand die weiteren Vertreter erlernen.

Kapitel 2

Die Vorbereitung: Diese Programme sind für das Programmieren in C++ nötig

Das Programmieren ist in gewisser Weise mit einer handwerklichen Tätigkeit vergleichbar: Bevor man sich ans Werk macht, ist es notwendig, die Arbeitsumgebung passend einzurichten. Außerdem müssen alle notwendigen Werkzeuge bereitstehen. Nur so ist es möglich, ein Programm zu erstellen. Wer mit dem Programmieren in C++ beginnen will, muss daher zunächst einige Programme auf seinem Computer installieren. Die folgenden Abschnitte stellen vor, um welche Software es sich dabei handelt und wie die Installation abläuft. Um in C++ zu programmieren, ist zunächst ein Texteditor notwendig. Dieser ermöglicht es, den Programmcode zu schreiben und im richtigen Format abzuspeichern. Um aus diesem Code ein lauffähiges Programm zu machen, ist ein Compiler erforderlich. Schließlich soll eine integrierte Entwicklungsumgebung (IDE) installiert werden. Diese übernimmt sowohl die Aufgaben des Compilers als auch des Texteditors. Außerdem bietet sie einige nützliche Zusatzfunktionen. Während die Programme zu Beginn mit dem Texteditor und mit dem Compiler erstellt werden, wird später die IDE als professionelle Alternative dazu eingeführt. Für dieses Buch kommen ausschließlich Programme zum Einsatz, die kostenfrei erhältlich sind. Daher fallen keine weiteren Ausgaben an.

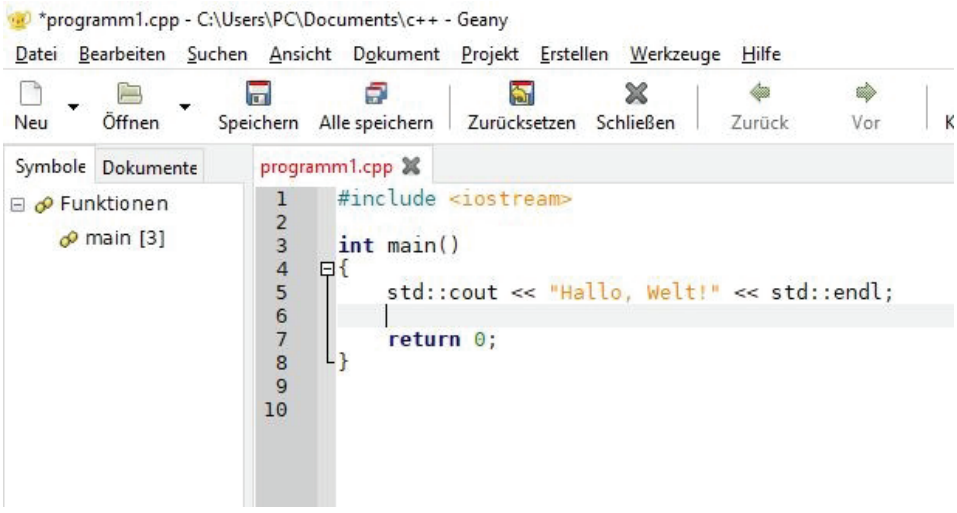
2.1 Ein Texteditor für die Gestaltung des Programmcodes

Wie bereits erwähnt, bestehen Computerprogramme zunächst lediglich aus Text. Um diesen zu erstellen, ist jedoch ein geeignetes Programm notwendig. Viele Menschen, die noch keine Erfahrungen im Bereich der Programmierung haben, denken hierbei an ein Textverarbeitungsprogramm wie Word. Dieses ist jedoch für diese Aufgabe un-

geeignet. Das liegt daran, dass diese Software nicht nur den Text abspeichert. Darüber hinaus fügt sie zahlreiche Informationen zu Gestaltung hinzu. Diese sind jedoch für das Computerprogramm unerheblich und stören die Ausführung.

Für diese Aufgabe ist es notwendig, einen Texteditor zu verwenden. Diese Programme speichern lediglich den Text ab. Die meisten Betriebssysteme verfügen bereits über einen integrierten Texteditor. Bei Windows ist beispielsweise das Programm Microsoft Editor – auch unter der Bezeichnung Notepad bekannt – integriert. Die Linux-Distribution Ubuntu wird mit gedit und die Distribution KDE mit Kate ausgeliefert. Leser, die eine der beiden genannten Linux-Distributionen verwenden, sind bereits bestens ausgerüstet und müssen keinen weiteren Texteditor installieren. Der Microsoft Editor eignet sich zwar im Prinzip ebenfalls für diese Aufgabe, doch ist sein Funktionsumfang nur minimal. Daher ist es empfehlenswert, einen etwas umfassenderen Texteditor zu installieren.

Dieser bietet viele Vorteile. Beispielsweise hebt er die Syntax eines Computerprogramms hervor und erstellt automatische Einrückungen. Das macht den Code übersichtlicher und verständlicher. Es ist möglich, Textbereiche, die im Moment nicht benötigt werden, einzuklappen und manche Editoren bieten sogar eine automatische Vervollständigung des Codes.



Screenshot 1 Die Syntax-Hervorhebung in einem Texteditor

Die Auswahl an Texteditoren ist sehr groß. Folgende Internetseite gibt einen kleinen Überblick darüber:

https://de.wikipedia.org/wiki/Liste_von_Texteditoren

Im Prinzip ist es möglich, fast jede dieser Möglichkeiten auszuwählen. In diesem Buch kommt der Texteditor Geany zum Einsatz. Für die hier gestellten Aufgaben ist es jedoch problemlos möglich, auch eine andere Alternative auszuwählen. Geany eignet sich jedoch sehr gut, da dieses Programm Code in C++ unterstützt und außerdem umfangreiche Funktionen für eine einfache Programmierung bietet. Darüber hinaus ist es kostenlos erhältlich. Es steht unter folgender Seite zum Download bereit:

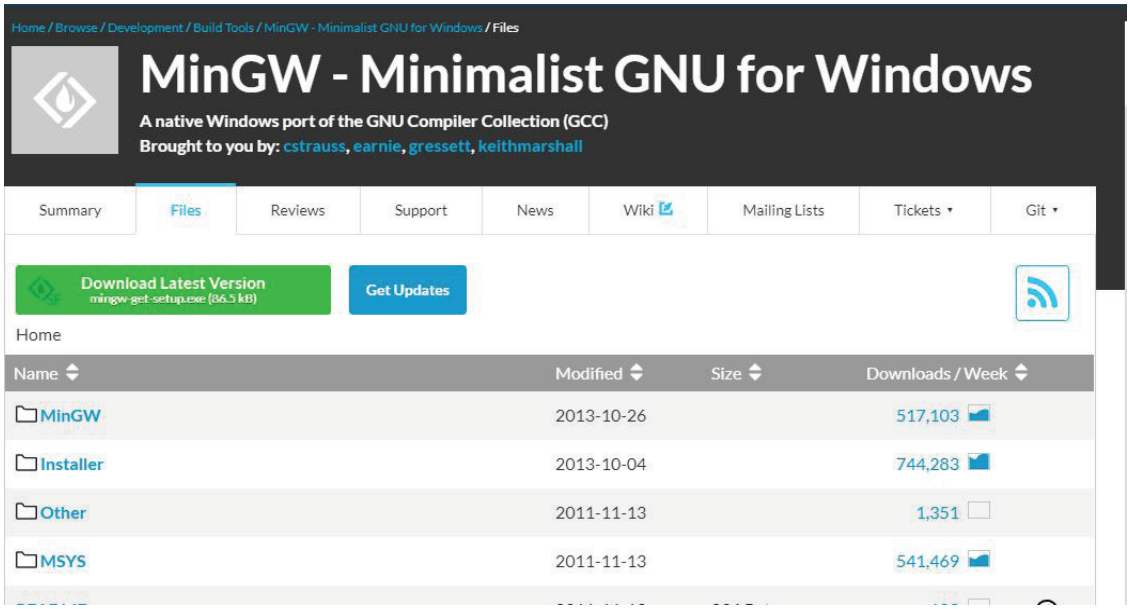
https://www.geany.org/Download/Releases

Dabei ist es lediglich notwendig, die Version für das entsprechende Betriebssystem herunterzuladen und anschließend mit der Standard-Konfiguration zu installieren.

2.2 Ein Compiler für die Erstellung der Programme

Um ein Programm in C++ auszuführen, ist ein Compiler notwendig. Dieser erzeugt aus dem Programmtext ein Programm in Maschinensprache. Wer Linux verwendet, genießt den Vorteil, dass hierbei in fast allen Distributionen bereits ein passender Compiler integriert ist. Bei einem Windows-Rechner ist es jedoch notwendig, diesen separat zu installieren. Hierfür gibt es mehrere Möglichkeiten. In diesem Buch soll der Compiler MinGW verwendet werden. Dieser steht unter folgender Seite kostenlos zum Download bereit:

https://sourceforge.net/projects/mingw/files/



Screenshot 2 Die Downloadseite für MinGW

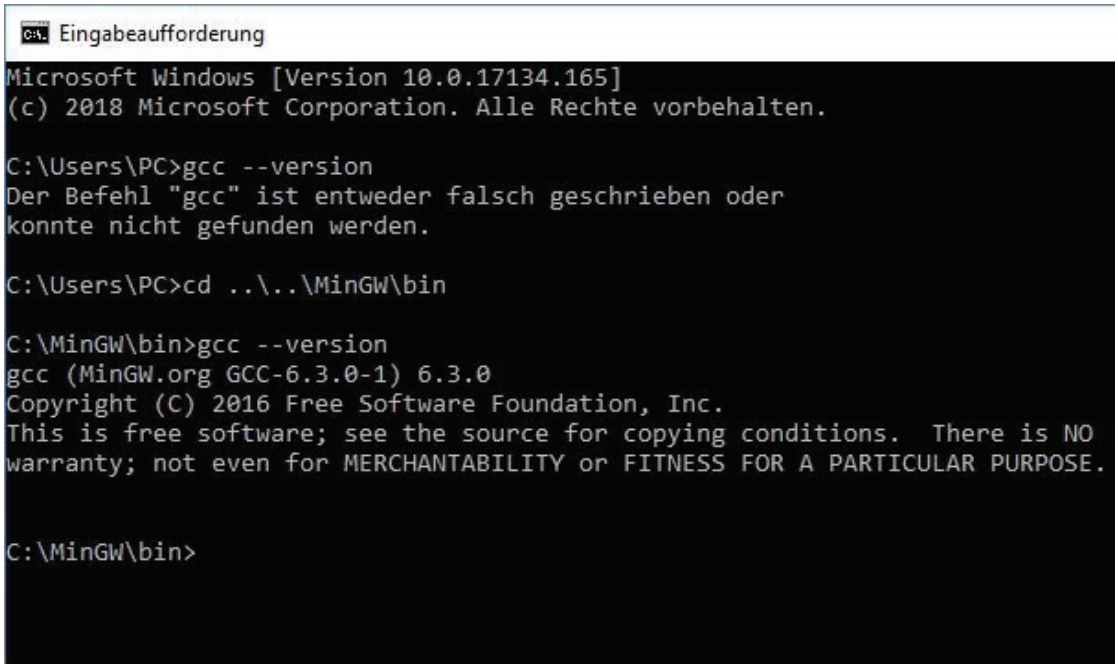
Nun ist es lediglich notwendig, das grüne Feld mit der neuesten Version anzuklicken, um den Installations-Assistenten herunterzuladen. Dieser muss danach geöffnet werden. Nach der Bestätigung der Standard-Konfiguration wird das Programm auf dem Rechner installiert.

Um dieses zu nutzen, ist es notwendig, einen Kommandozeileninterpreter zu verwenden. Diese ist unter Windows ganz einfach aufzufinden, indem man den Begriff cdm in das Suchfeld eingibt. Daraufhin erscheint das entsprechende Programm sofort. Alternativ dazu ist es möglich, es über das Startmenü aufzurufen: im Ordner Windows-System unter dem Begriff Eingabeaufforderung.

Wenn die Kommandozeile geöffnet ist, ist es empfehlenswert, zu überprüfen, ob die Installation erfolgreich abgeschlossen wurde. Dazu ist es sinnvoll, die Version abzufragen. Das geschieht durch die Eingabe des Begriffs gcc --version. In der Regel erscheint dabei jedoch zunächst folgende Fehlermeldung: Der Befehl “gcc” ist entweder falsch geschrieben oder konnte nicht gefunden werden.

Das liegt daran, dass es nur möglich ist, das Programm zu verwenden, wenn man sich im Verzeichnis befindet, in dem es installiert wurde. Wenn bei der Installation hierfür das Standardverzeichnis gewählt wur-

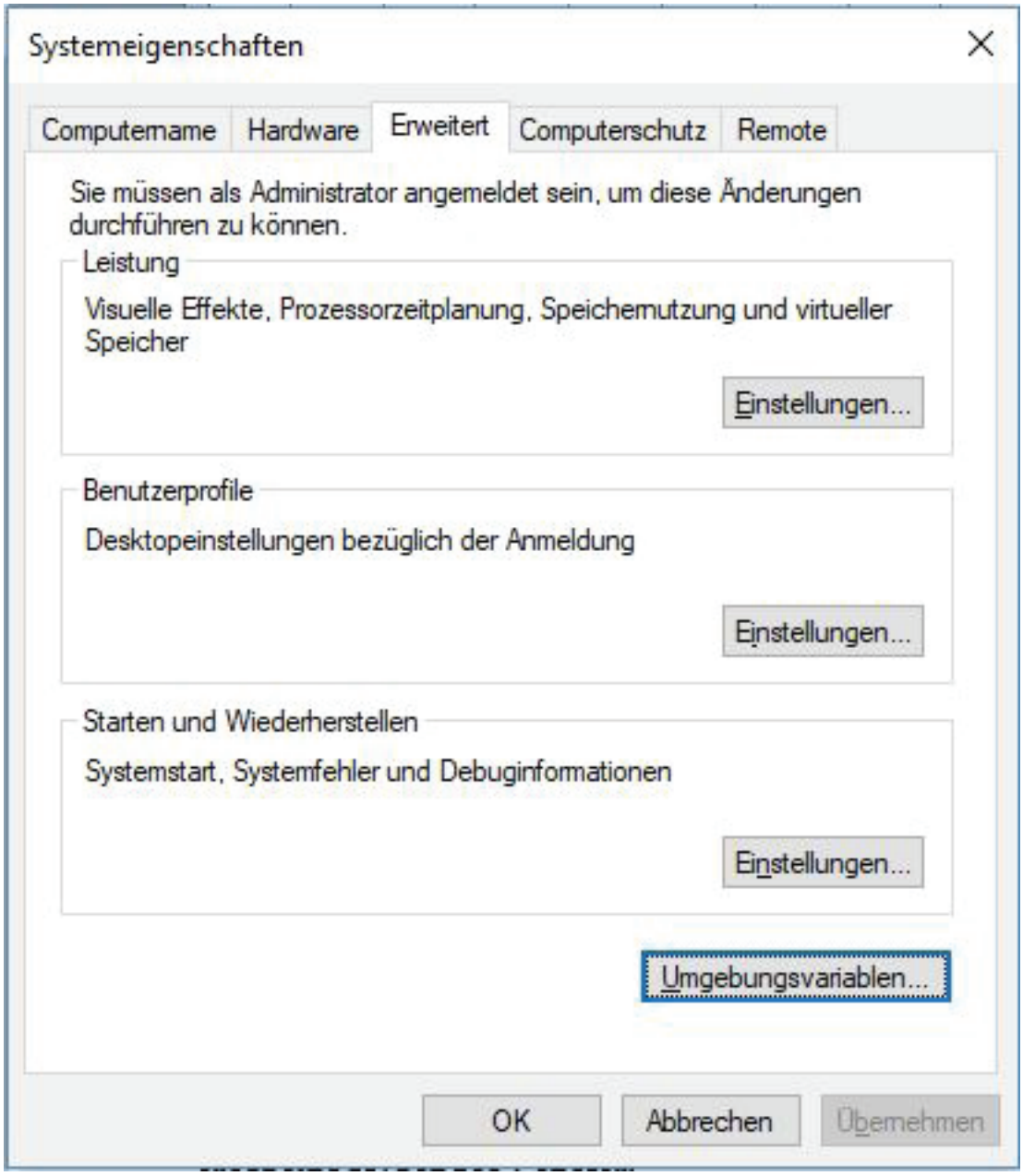
de, sollte dies C:\MinGW\bin sein. Falls das Programm an einem anderen Ort abgelegt wurde, ist es notwendig, den entsprechenden Pfad einzugeben und stets mit dem Zusatz \bin zu versehen. Danach sollte bei der Wiederholung des Befehls die entsprechende Version angezeigt werden.



Screenshot 3 Die Versionsabfrage – zunächst mit der Fehlermeldung und anschließend erfolgreich im richtigen Verzeichnis.

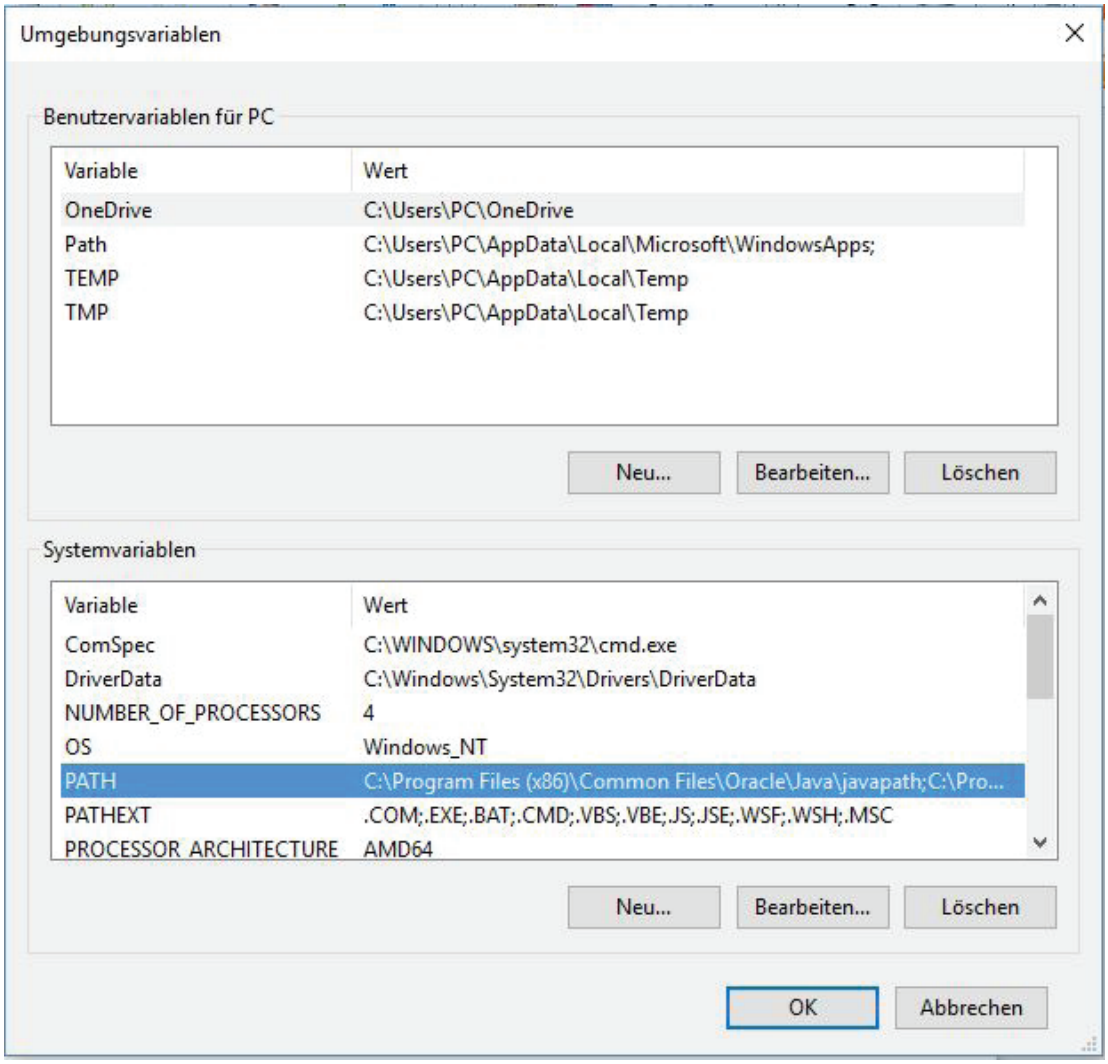
Daraus wird deutlich, dass das Programm MinGW bisher nur in dem Verzeichnis erreichbar ist, in dem es installiert wurde. Das gilt nicht nur für die Versionsabfrage, sondern auch für die Kompilierung der Dateien. Das würde den großen Nachteil mit sich bringen, dass alle Programme, die erstellt werden sollen, ebenfalls im Installationsverzeichnis abgelegt werden müssten. Da dies ausgesprochen unübersichtlich wäre, ist es sinnvoll, MinGW auch aus anderen Verzeichnissen zugänglich zu machen. Dazu ist es notwendig, die Umgebungsvariablen anzupassen.

Hierfür müssen die erweiterten Systemeinstellungen aufgerufen werden – entweder über die Systemeinstellungen in der Startleiste oder direkt über die Windows-Suchfunktion. Daraufhin erscheint folgendes Fenster:



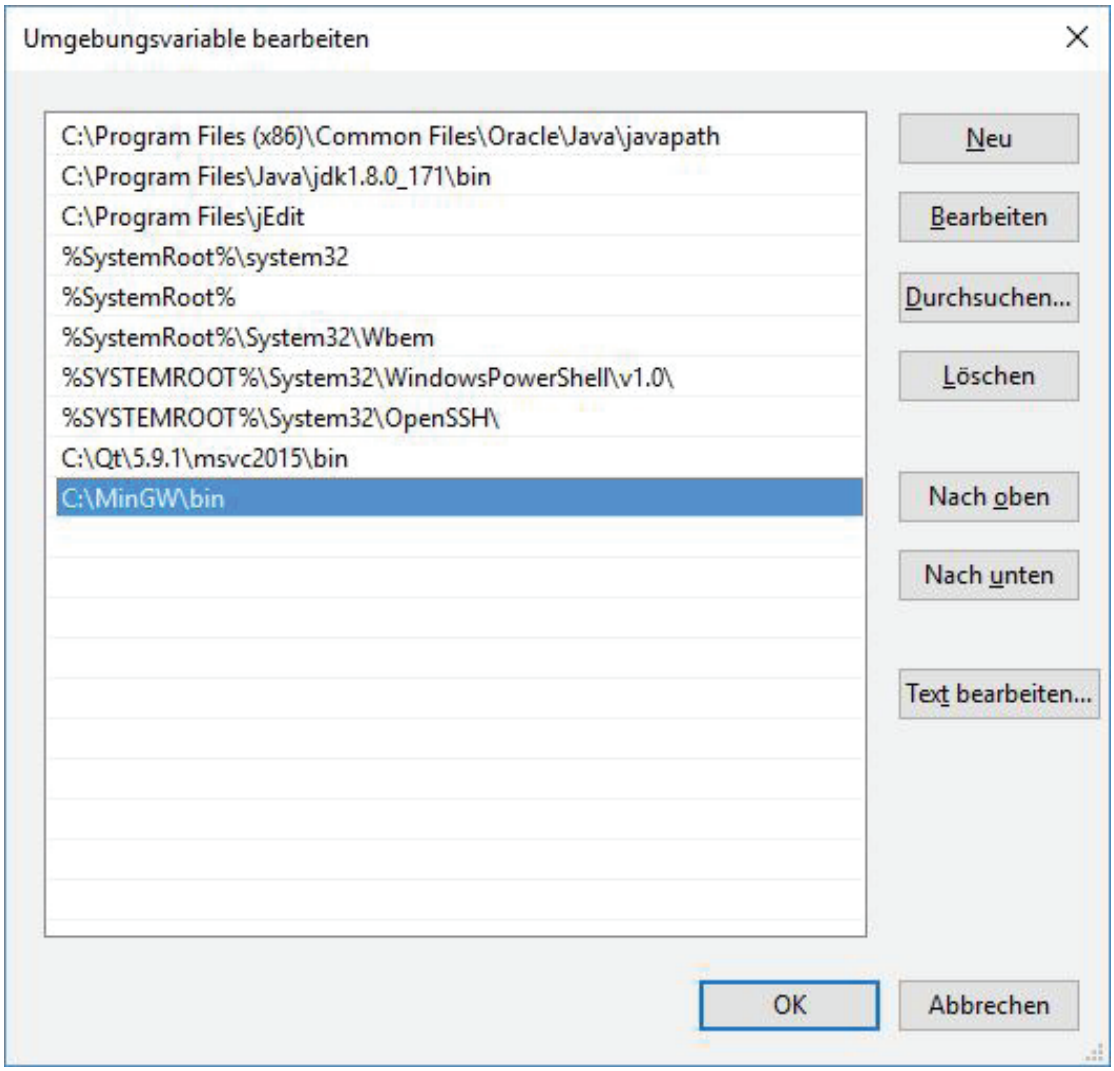
Screenshot 4 Die erweiterten Systemeinstellungen

Im unteren Bereich befindet sich die Schaltfläche “Umgebungsvariablen”. Nachdem diese angeklickt wurde, erscheint folgendes Fenster:



Screenshot 5 Das Fenster für die Einstellung der Umgebungsvariablen

Nun ist es notwendig, im unteren Bereich die Zeile mit der Bezeichnung PATH anzuklicken und anschließend "Bearbeiten" auszuwählen.

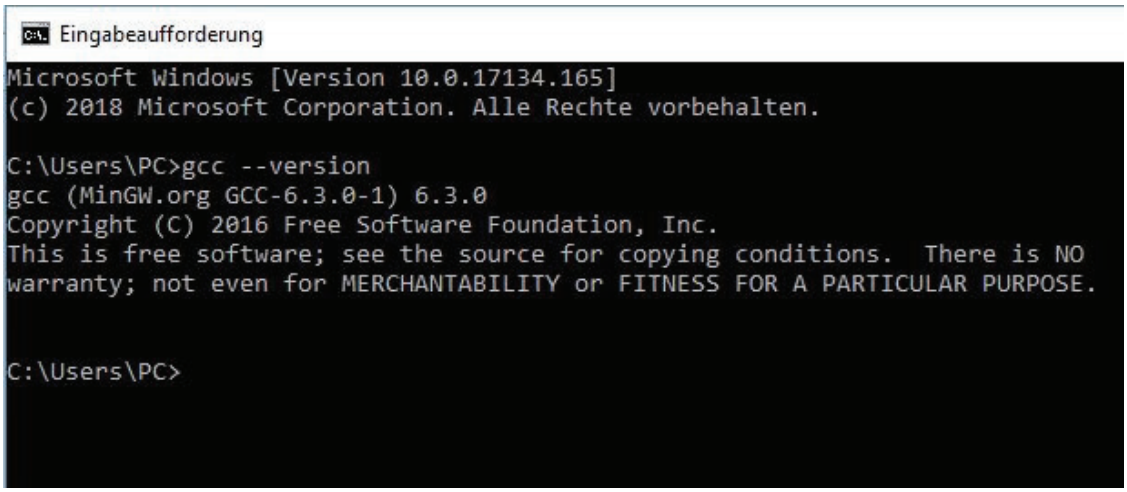


Screenshot 6 Eine neue Umgebungsvariable hinzufügen

Daraufhin öffnet sich das Fenster, das im obigen Screenshot zu sehen ist. Durch das Anklicken von "Neu" ist es möglich, eine neue Umgebungsvariable hinzuzufügen. Diese muss den Pfad enthalten, der zu MinGW führt – bei der Installation im Standard-Verzeichnis also C:\MinGW\bin.

Nun ist es nur noch notwendig, alle offenen Fenster mit "OK" zu bestätigen, um MinGW auch aus anderen Verzeichnissen zugänglich zu machen. Damit die Änderungen wirksam werden, ist es jedoch notwendig, den Kommandozeileninterpreter zunächst zu schließen, falls er noch geöffnet sein sollte. Wenn er erneut geöffnet wird, soll-

te die Versionskontrolle in jedem beliebigen Verzeichnis erfolgreich verlaufen.



Screenshot 7 Nun ist die Versionskontrolle auch in anderen Verzeichnissen erfolgreich.

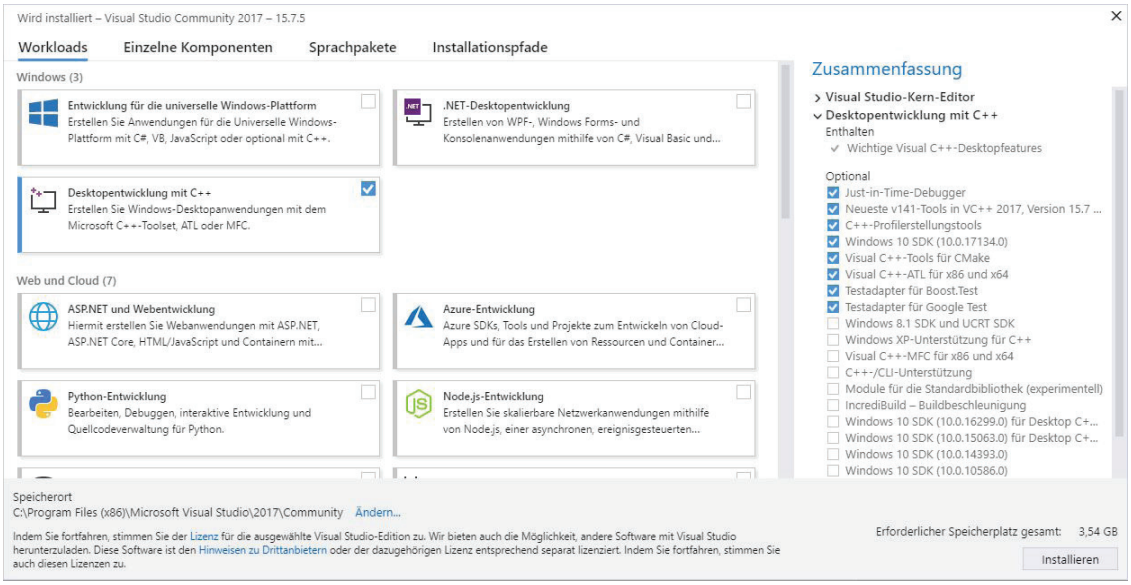
2.3 Visual Studio: eine integrierte Entwicklungsumgebung

Abschließend soll noch eine integrierte Entwicklungsumgebung (Integrated Development Environment – IDE) installiert werden. Dabei handelt es sich um ein Programm, das die Funktionen von Texteditor und Compiler miteinander vereint. Außerdem bietet es viele nützliche Zusatzfunktionen. Professionelle Programmierer arbeiten meistens mit einer IDE. Dieses Programm wird erst ab Kapitel 12 verwendet. Wer schon ungeduldig ist und möglichst schnell mit dem Programmieren beginnen will, kann die Installation daher noch etwas verschieben. Sie sollte dann jedoch vor dem zwölften Kapitel nachgeholt werden.

Visual Studio ist eine sehr hochwertige Entwicklungsumgebung des Software-Konzerns Microsoft. Dabei gibt es eine kostenpflichtige Ausführung sowie die Gratis-Version Visual Studio Community Edition. Die kostenlose Version ist für diesen Kurs vollkommen ausreichend. Sie steht unter folgender Adresse zum Download bereit:

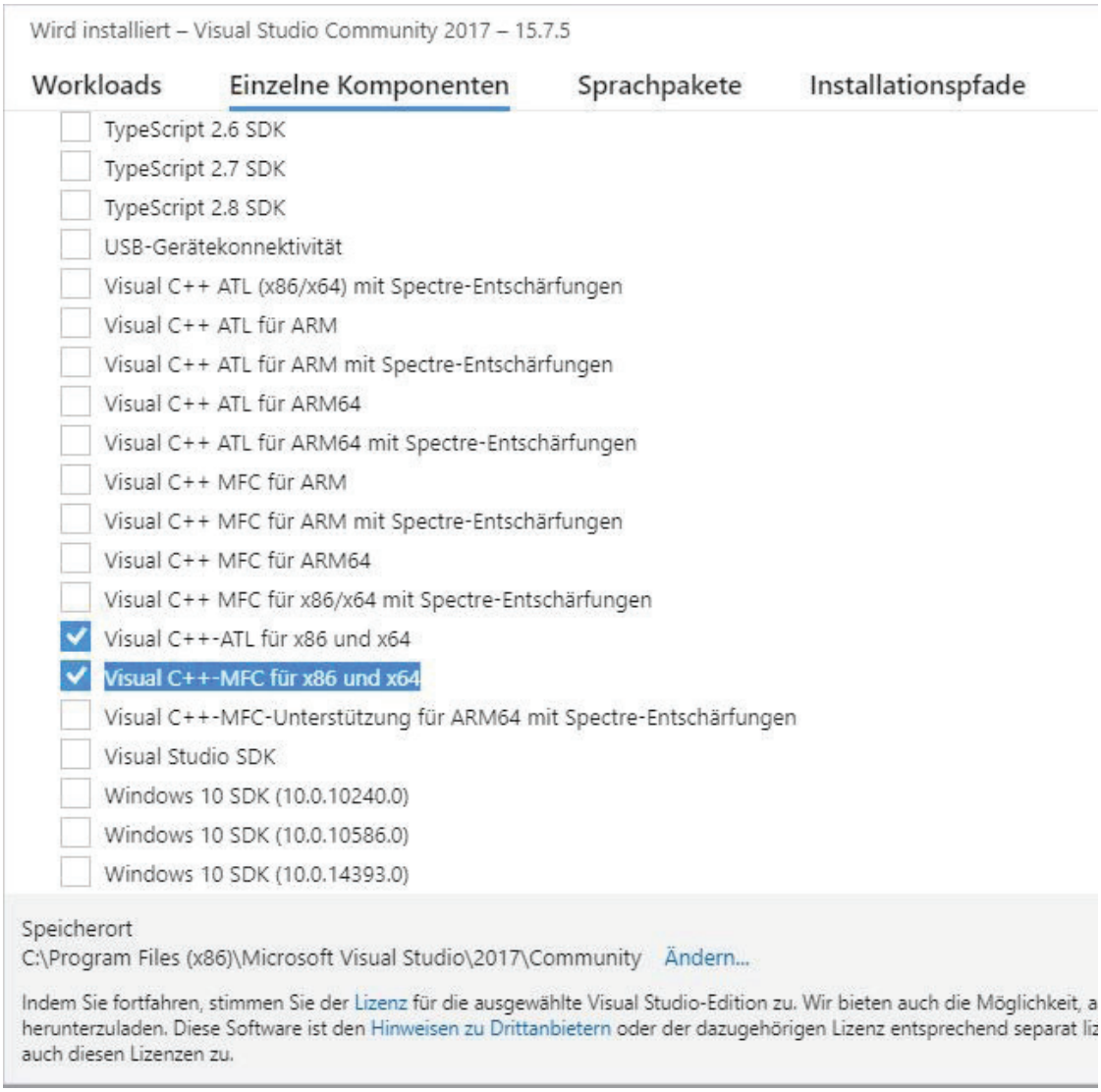
https://visualstudio.microsoft.com/de/vs/

Der Installations-Assistent, der hier zur Verfügung steht, macht die Installation des Programms ganz einfach. Allerdings ist es wichtig, bei der Auswahl der Komponenten, die installiert werden sollen, die richtigen Bereiche anzuklicken.



Screenshot 8 Die Auswahl der Desktopentwicklung mit C++

Besonders wichtig ist dabei die Desktopentwicklung mit C++. Danach ist es notwendig, den Reiter “Einzelne Komponenten” anzuklicken. Hier ist eine lange Liste mit verschiedenen Einträgen vorhanden. Dabei ist es notwendig, weit nach unten zu scrollen. Unter der Überschrift “SDKs, Bibliotheken und Frameworks” befindet sich der Eintrag “Visual C++-MFC für x86 und x64”. Dieser muss angeklickt werden. Dabei ist es wichtig, alle anderen Einstellungen nicht zu verändern. Nun ist es nur noch notwendig, die Schaltfläche “Installieren” zu betätigen.



Screenshot 9 Die Auswahl von MFC

Dieser letzte Schritt fügt der Software MFC – Microsoft Foundation Classes – hinzu. Diese sind für die Erstellung grafischer Benutzeroberflächen wichtig und kommen in Kapitel 13 zum Einsatz. Visual Studio steht nur für Windows und für MacOS zur Verfügung. Leser, die ausschließlich das Betriebssystem Linux verwenden, können daher die Aufgaben in den entsprechenden Kapiteln leider nicht bearbeiten.

Kapitel 3

Das erste eigene Programm in C++ schreiben

Nachdem nun ein erster Überblick über die Eigenschaften der Programmiersprache C++ gegeben wurde und alle notwendigen Programme installiert sind, ist es an der Zeit, selbst Hand anzulegen. Das erste Programm ist dabei selbstverständlich sehr einfach aufgebaut und bietet nur eine einzige Funktion. Dennoch ist es wichtig, alle Details dieses Programms genau zu besprechen. Daran wird die grundlegende Struktur eines C++-Programms ersichtlich, die den Ausgangspunkt für alle weiteren Aufgaben darstellt.

3.1 Der erste Schritt: eine einfache Ausgabe auf dem Bildschirm

Das Ziel des ersten Programms soll es lediglich sein, eine kurze Textnachricht auszugeben. Bei modernen Anwenderprogrammen erfolgt eine derartige Ausgabe in der Regel in einem eigenen Fenster. Um diese Benutzeroberflächen zu erstellen, sind jedoch bereits etwas fortgeschrittenere Programmierkenntnisse erforderlich. Daher wird auf diese Möglichkeit erst in Kapitel 13 eingegangen. Zu Beginn soll eine einfachere Form der Ausgabe gewählt werden: über den Kommandozeileninterpreter.

Um das Programm zu schreiben, ist es notwendig, den Texteditor zu öffnen und den entsprechenden Programmcode einzugeben. Der wesentliche Befehl zur Ausgabe eines Texts lautet:

```
std::cout << "Ausgabetext";
```

Allerdings muss dieser Befehl in die grundlegende Struktur eines C++-Programms eingebunden werden. Wenn das Programm eine klei-

ne Begrüßung zum C++-Kurs ausgeben soll, ergibt sich folgender Programmcode:

```
#include <iostream>

int main ()
{
      std::cout << "Willkommen zum C++-Kurs!";
}
```


Nun ist es nur noch notwendig, das Programm abzuspeichern. Dabei ist es möglich, den Namen frei zu wählen. Es ist lediglich notwendig, die Endung .cpp hinzuzufügen. Diese kennzeichnet ein C++-Programm. Da es sich beim ersten Programm um einen Willkommensgruß handelt, soll es den Namen willkommen.cpp erhalten.

3.2 Die Elemente des Programmcodes verstehen

Um den Aufbau und die Funktionsweise eines C++-Programms zu verstehen, ist es wichtig, die einzelnen Bestandteile genau zu erläutern. Zunächst soll dabei der eigentliche Befehl und anschließend die Struktur des Programms erklärt werden.

Der Befehl für die Ausgabe lautet `cout`. Allerdings kann dieser nicht für sich alleine stehen. Das liegt daran, dass alle Befehle in C++ in sogenannten Bibliotheken abgespeichert sind. Darin ist festgehalten, welche Aktion der Compiler durchführen soll, wenn der entsprechende Ausdruck im Programm auftaucht. Allerdings gibt es sehr viele unterschiedliche Bibliotheken, die der Compiler standardmäßig unterstützt. Darüber hinaus kann jeder Anwender sogar seine eigene Bibliothek entwerfen. Dabei ist es möglich, dass ein bestimmter Ausdruck in mehreren Bibliotheken auftaucht. Bei einer doppelten Belegung wüsste der Compiler daher nicht, welche Funktionen er anwenden soll. Aus diesem Grund arbeitet C++ mit sogenannten Namensräumen. Darin darf jeder Befehl nur ein einziges Mal bestimmt werden. Daher muss der Programmierer

vor jedem Befehl bestimmen, zu welchem Namensraum er gehört. Um diesen anzugeben, kommt das Kürzel `std` zum Einsatz. Dieses steht für Standard. Der zugehörige Namensraum umfasst alle Standard-Befehle in C++. Um den Befehl mit dem Namensraum zu verbinden, kommt der Zuweisungsoperator `::` zum Einsatz. Schließlich folgt der Text, der ausgegeben werden soll. Dieser muss immer in doppelten Anführungszeichen stehen. Durch die doppelten Pfeile (<<) wird dieser Inhalt dem Ausgabebefehl zugewiesen. Am Ende des Befehls muss immer ein Semikolon stehen.

Zu Beginn des Programmcodes steht der Befehl `#include <iostream>`. Wie soeben beschrieben, sind die einzelnen Befehle, die für das Programm zum Einsatz kommen, in Bibliotheken abgespeichert, die der Compiler beim kompilieren einbinden muss. Das macht er jedoch nicht automatisch. Es ist notwendig, ihm die einzelnen Dateien, die er verwenden soll, mitzuteilen. Das geschieht über den Befehl `#include`. Der Befehl, der für dieses Programm verwendet wird, befindet sich in der Datei `iostream`. Daher muss deren Name in spitzen Klammern eingefügt werden.

Danach wird das eigentliche Programm geöffnet. Der Schlüsselbegriff hierfür ist `main`. Dieser öffnet die Hauptfunktion eines Programms und stellt den Einstiegspunkt dar. In C++ kann jede Funktion einen Wert zurückgeben und außerdem ist es möglich, Daten an die Funktion zu übergeben. Das ist bei diesem Programm zwar nicht notwendig, doch wirkt es sich auf dessen Struktur aus. Vor dem Begriff `main` muss daher der Datentyp des Rückgabewerts eingefügt werden. Daher steht hier `int` – eine Abkürzung für integer beziehungsweise auf Deutsch für eine ganze Zahl. Den Wert, der hierbei zurückgegeben wird, kann der Programmierer selbst bestimmen. Er dient in der Regel als Hinweis darauf, ob das Programm erfolgreich abgelaufen ist. Enthält das Programm keinen spezifischen Befehl zum Rückgabewert, gibt es stets den Wert 0 zurück.

In der Klammer stehen die Werte, die der Funktion übergeben werden sollen. Das ist hier jedoch nicht notwendig, sodass die Klammer leer bleibt. An diesem Punkt ist es noch nicht notwendig, die genaue Funktionsweise der Rück- und Übergabewerte zu verstehen. Es ist lediglich wichtig, zu beachten, dass C++-Programme in der Regel mit der Zeile `int main ()` beginnen.

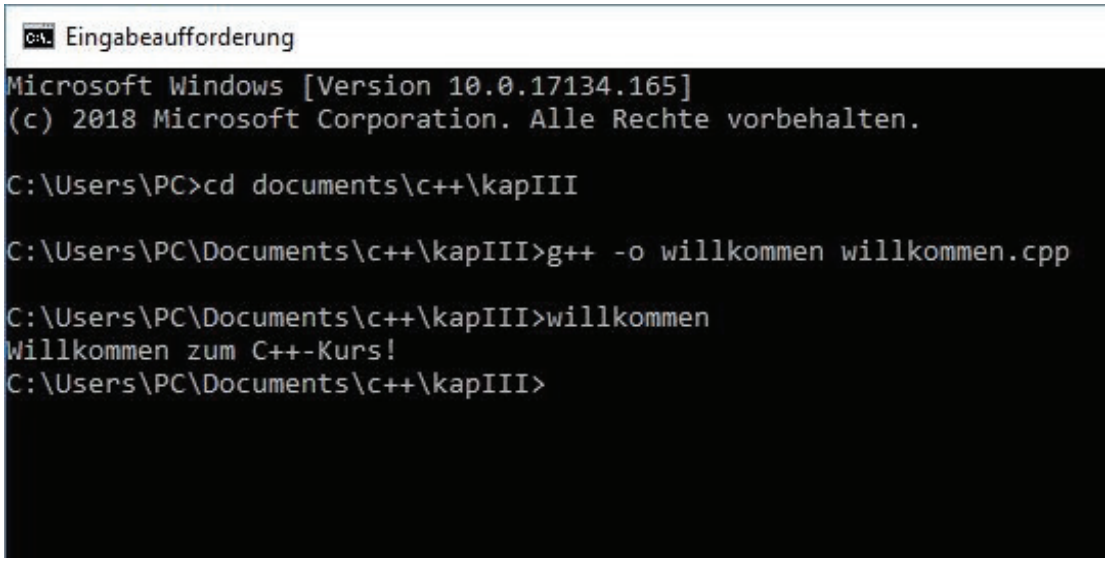
Der eigentliche Inhalt des Programms steht schließlich in geschweiften Klammern. Diese markieren den Anfang und das Ende und es ist wichtig, sie auf keinen Fall zu vergessen.

3.3 Das Programm ausführen

Der letzte Schritt besteht darin, das Programm zu kompilieren und auszuführen. Hierfür ist wieder der Kommandozeileninterpreter notwendig, dessen Verwendung bereits bei der Installation des Compilers erklärt wurde. Dafür ist es erforderlich, diesen zu öffnen und anschließend in das Verzeichnis zu wechseln, in dem das Programm abgespeichert wurde. Anschließend muss folgender Befehl eingegeben werden:

```
g++ -o willkommen willkommen.cpp
```

Wenn alles nach Plan verläuft, entsteht eine Wartezeit von wenigen Sekunden und danach erscheint wieder die Zeile für die Eingabe. Auf den ersten Blick scheint es, als wäre nichts passiert. Dieser Eindruck trügt jedoch. Wenn man den sich den Inhalt des entsprechenden Verzeichnisses betrachtet (entweder über das Fenster des Windows-Dateimanagers oder in der Kommandozeile über den Befehl dir), fällt auf, dass hier nun die Datei willkommen.exe entstanden ist. Dabei handelt es sich um eine ausführbare Datei, die man durch die Eingabe des Befehls willkommen aufrufen kann. Danach erscheint der Text, der im Programm für die Ausgabe angegeben wurde, im Kommandozeileninterpreter:



Screenshot 10 Das Kompilieren und die Ausgabe des ersten Programms

Nachdem das erste C++-Programm erfolgreich kompiliert und ausgeführt wurde, ist es sinnvoll, sich nochmals die entsprechenden Befehle genau anzuschauen. Um ein C++-Programm im Kommandozeileninterpreter zu kompilieren, kommt stets der Befehl g++ zum Einsatz. Zum Schluss steht der Name der Datei, die kompiliert werden soll. Während diese beiden Bestandteile unbedingt notwendig sind, ist der Mittelteil optional. Um dessen Funktion herauszufinden, ist es möglich, das Programm erneut zu kompilieren – dieses Mal jedoch nur mit dem Befehl g++ willkommen.cpp. Wenn man nun den Inhalt des entsprechenden Verzeichnisses abfragt, ist eine neue Datei mit dem Namen a.exe entstanden. Deren Funktion ist genau die gleiche wie bei der zuvor erstellten Datei willkommen.exe. Der Einschub -o dient dazu, den Namen der neu zu erstellenden exe-Datei vorzugeben. Fehlt diese Angabe, erstellt der Compiler automatisch eine Datei mit dem Namen a.exe. Der Name der exe-Datei kann dabei frei gewählt werden und muss nicht zwingend mit dem Namen der cpp-Datei übereinstimmen. Um die Übersicht zu behalten ist es jedoch ratsam, stets die gleichen Bezeichnungen zu verwenden.

Um das Programm auszuführen, ist es lediglich notwendig, den Namen der kompilierten Datei einzugeben – allerdings ohne die Endung .exe.

3.4 Übungsaufgabe: So lässt sich der Programmcode verändern

Am Ende der meisten Kapitel dieses Buchs stehen einige Übungsaufgaben. Diese dienen dazu, das Gelernte zu vertiefen und selbst anzuwenden. In der Regel sind die Befehle, die in den vorhergehenden Kapiteln vorgestellt wurden, für die Bearbeitung ausreichend. Sollten dafür zusätzliche Kenntnisse notwendig sein, werden die entsprechenden Details kurz in der Aufgabenstellung erläutert.

Häufig entsprechen die Programme, die in den Übungsaufgaben erstellt werden, in einigen Teilen den Beispielen der vorhergehenden Kapitel. Daher wäre es möglich, die entsprechenden Programme lediglich zu kopieren und daraufhin die Funktionen abzuändern. Um sich die grundlegenden Strukturen besser einzuprägen, wird jedoch unbedingt dazu geraten, den Code von Grund auf neu zu schreiben.

Abschließend werden die Lösungen präsentiert. In manchen Fällen – insbesondere bei fortgeschrittenen Aufgaben – sind mehrere Lösungswege möglich. Die Musterlösungen dienen daher nur als Abgleich und als Hilfestellung, wenn man selbst nicht weiter kommt. Wenn das eigene Programm die Anforderungen der Aufgabenstellung erfüllt, obwohl es einen anderen Programmcode als die vorgestellte Lösung verwendet, ist die Aufgabe ebenfalls korrekt bearbeitet. Nach diesen grundsätzlichen Erläuterungen können Sie mit den Aufgaben für dieses Kapitel beginnen:

1. Erweitern Sie das Programm aus diesem Kapitel so, dass es einen zweiten Ausgabebefehl enthält. Den Inhalt können Sie beliebig festlegen.
2. Wenn Sie das Programm aus Aufgabe 1 ausführen, stellen Sie fest, dass dabei beide Textbausteine ohne Zeilenumbruch hintereinander ausgegeben werden. Fügen Sie daher nach der ersten Textausgabe den Begriff `std::endl` ein, um einen Zeilenumbruch zu erzeugen. Diesen müssen Sie einfach durch die doppelten Pfeile (<<) an den bisherigen Befehl anschließen.

3. Im letzten Programm kamen mehrere Befehle aus dem Namensraum `std` zum Einsatz. Diesen Zusatz jedes Mal hinzuzufügen, ist jedoch recht umständlich. Wenn sie in einem Programm stets den gleichen Namensraum verwenden, ist es anstatt dessen möglich, dies durch den Befehl `using namespace std;` zu Beginn des Programms anzugeben. Fügen Sie diesen Befehl ein und entfernen Sie die übrigen Angaben zum Namensraum aus dem Programm.

Lösungen:

1.

```
#include <iostream>

int main ()
{
      std::cout << "Willkommen zum C++-Kurs!";
      std::cout << "Hier erlernen Sie die Grundlagen der
      Programmiersprache C++.";
}
```

2.

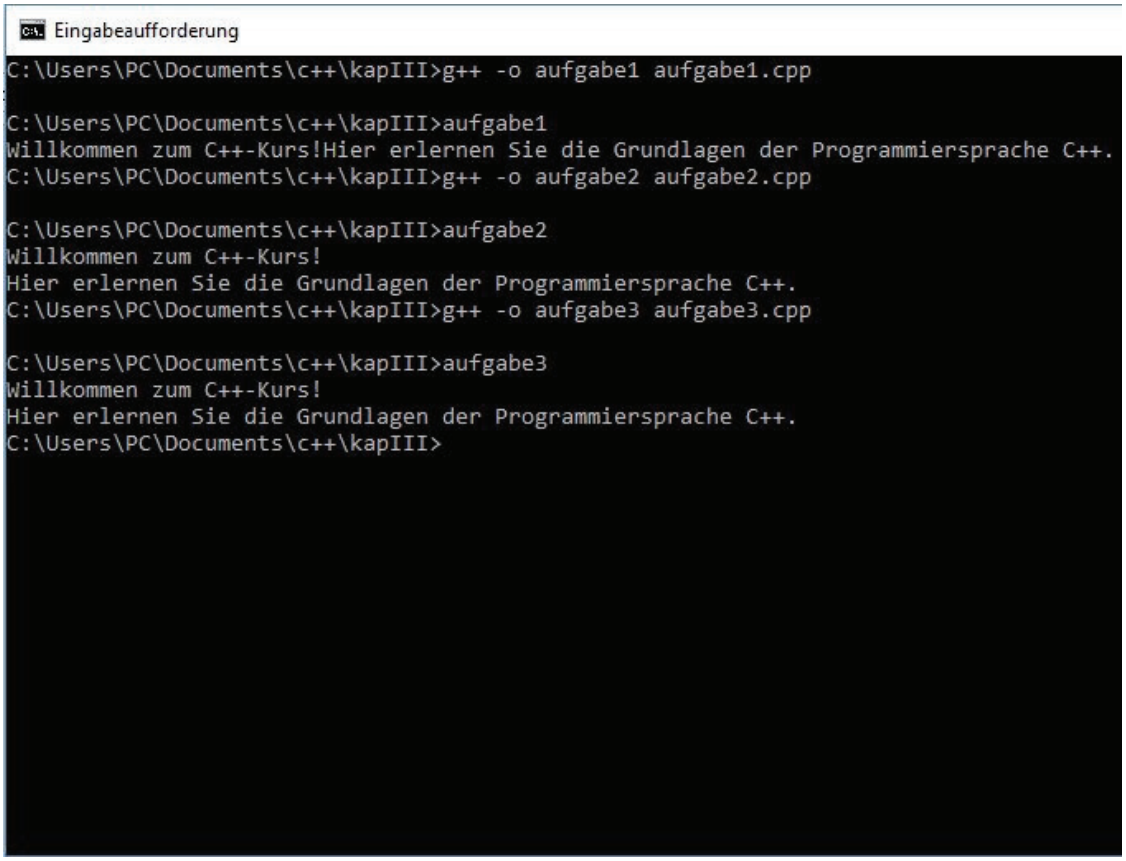
```
#include <iostream>

int main ()
{
      std::cout << "Willkommen zum C++-Kurs!" << std::endl;
      std::cout << "Hier erlernen Sie die Grundlagen der
      Programmiersprache C++.";
}
```

3.

```
#include <iostream>
using namespace std;

int main ()
{
      cout << "Willkommen zum C++-Kurs!" << endl;
      cout << "Hier erlernen Sie die Grundlagen der Programmiersprache
      C++.";
}
```



Screenshot 11 Die Kompilierung und die Ausgabe der drei Programme

Hat Ihnen diese Leseprobe gefallen?

Bestellen Sie das Buch als Taschenbuch komfortabel auf Amazon!

eBook: https://amzn.to/2GLG024

Taschenbuch: https://amzn.to/2T1J3JK



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/