Aktuell zu C++11

C++

Programmieren mit einfachen Beispielen

DIRK LOUIS

Markt+Technik

Visual C++ 2010 Express Edition, C++-Referenz, Beispiele

CD

ALWAYS LEARNING

PEARSON

C++

C++

Programmieren mit einfachen Beispielen

DIRK LOUIS

Markt+Technik

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.

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Fast alle Hardware- und Softwarebezeichnungen und weitere Stichworte und sonstige Angaben, die in diesem Buch verwendet werden, sind als eingetragene Marken geschützt. Da es nicht möglich ist, in allen Fällen zeitnah zu ermitteln, ob ein Markenschutz besteht, wird das ® Symbol in diesem Buch nicht verwendet.

10 9 8 7 6 5 4 3 2 1

13 12

ISBN 978-3-8272-4750-6

© 2012 by Markt+Technik Verlag,
ein Imprint der Pearson Deutschland GmbH,
Martin-Kollar-Straße 10–12, D-81829 München/Germany
Alle Rechte vorbehalten
Covergestaltung: Thomas Arlt, tarlt@adesso21.net
Titelfoto: Getty Images
Lektorat: Brigitte Bauer-Schiewek, bbauer@pearson.de
Korrektorat und Fachlektorat: Petra Alm
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: Ulrich Borstelmann, Dortmund (www.borstelmann.de)
Druck und Verarbeitung: Drukarnia Dimograf, Bielsko-Biala
Printed in Poland

Kapitel 3

Wie erstellt man eigene Programme?

In diesem Kapitel geht es darum, dass Sie Ihren Compiler und Ihre Programmierumgebung wählen und kennen lernen. Lesen Sie dieses Kapitel aufmerksam durch, denn die beschriebenen Techniken brauchen Sie zur Nachprogrammierung der in den folgenden Kapiteln vorgestellten Programme.

Welcher Compiler darf es sein?

Stellvertretend für viele andere Compiler stelle ich Ihnen hier zwei äußerst zuverlässige und bewährte C++-Compiler vor:

- den **Visual C++-Compiler** für die Familie der **Windows**-Betriebssysteme

  Den Visual C++-Compiler können Sie in der Express-Edition kostenlos aus dem Internet herunterladen oder von der Buch-CD installieren (siehe Abschnitt »Programmerstellung mit Visual C++ (Windows)«).

- den **GNU-C++-Compiler** für **Linux**

  Wenn Sie mit Linux arbeiten, wird der GNU-C++-Compiler möglicherweise bereits auf Ihrem System installiert sein. Wenn nicht, sollten Sie ihn zumindest auf der Linux-Installations-DVD finden (siehe Abschnitt »Programmerstellung mit dem g++-GNU-Compiler«.)

Vielleicht möchten Sie aber auch mit einem ganz anderen Compiler arbeiten? Kein Problem! Sie können dieses Buch zusammen mit jedem Compiler lesen, der sich an den C++-ISO-Standard hält – was für die meisten gängigen Compiler erfreulicherweise zutrifft.

> **Hinweis**
>
> Die Installation der verschiedenen Compiler ist nicht wirklich schwierig, hat aber so ihre Tücken. Sollte es nicht auf Anhieb klappen, versuchen Sie es einfach noch einmal und achten Sie darauf, alle Schritte korrekt nachzuvollziehen. Lesen Sie im Zweifelsfall die Installationshinweise zu Ihrem Compiler. Sollte dies nicht fruchten, können Sie sich selbstverständlich auch an mich wenden. Ferndiagnosen zu fehlgeschlagenen Installationsversuchen sind zwar meist schwierig, aber soweit ich kann, helfe ich gerne.

Programmerstellung mit Visual C++ (Windows)

Als Beispiel für einen leistungsfähigen und zuverlässigen Compiler samt integrierter Entwicklungsumgebung für die Windows-Plattform möchte ich Ihnen den Visual C++-Compiler vorstellen, den Sie auch auf Ihrer Buch-CD finden.

Hinweis

Die Visual C++ Express Edition ist kostenfrei, muss aber nach 30 Tagen registriert werden. (Zumindest war dies die Politik Microsofts zum Zeitpunkt der Drucklegung dieses Buches.) Wenn Sie lieber mit einem Public License-Compiler arbeiten möchten, laden Sie sich für Windows MinGW herunter (`http://sourceforge.net/projects/mingw/`) und verwenden Sie die Aufrufbefehle, die weiter unten im Abschnitt zum GNU-Compiler beschrieben sind. (MinGW ist eine Windows-Hülle zu dem GNU-G++-Compiler.)

Hinweis

Sollten Sie die Buch-CD verloren haben, können Sie den Compiler auch von der Website `http://www.microsoft.com/germany/express/download` herunterladen. (Achtung! Links zu Webseiten können sich schnell ändern. Wenn der oben angegebene Link ins Leere führen sollte, suchen Sie mit Ihrer Suchmaschine nach den Begriffen »Visual C++«, »Express« und »Download«.)

Installation

1 Schließen Sie alle Programme und legen Sie die Buch-CD in Ihr DVD-Laufwerk ein.

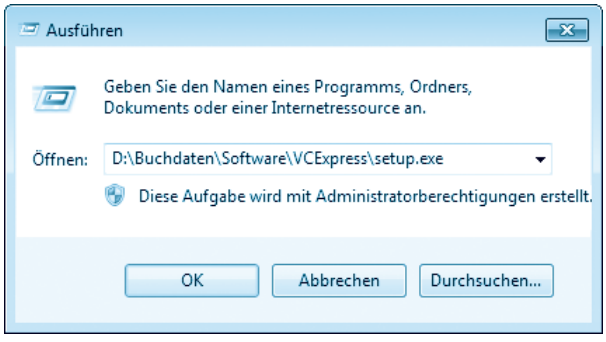

Abbildung 3.1: Start des Installationsprogramms über Start/Alle Programme/Zubehör/Ausführen (bzw. Start/Ausführen)

2 Führen Sie die EXE-Datei des Installationsprogramms aus, um den Compiler zu installieren.

Öffnen Sie beispielsweise das Start-Menü und wählen Sie unter *Alle Programme/Zubehör* den Befehl *Ausführen* (unter Windows XP steht der Befehl

direkt im Start-Menü). Im gleichnamigen Dialogfeld tippen Sie den Pfad zum Installationsprogramm ein oder wählen die *setup.exe*-Datei über den Schalter *Durchsuchen* und den zugehörigen Suchdialog aus. Schicken Sie den Dialog mit einem Klick auf den *OK*-Schalter ab.

Es erscheinen jetzt nacheinander das Begrüßungsfenster des VC-Installations-Assistenten, der Lizenzvertrag und die Abfrage der Installationsoptionen. (Die Option *Silverlight* können Sie getrost deaktivieren. Wir werden Silverlight nicht benötigen.) Klicken Sie jeweils auf *Weiter*, um mit der Installation fortzufahren. Im letzten Fenster sehen Sie dann noch einmal eine Zusammenfassung. Klicken Sie dann auf *Installieren*.

Abbildung 3.2: Wundern Sie sich nicht, wenn Ihnen in der Zusammenfassung nur »Microsoft Visual Express« angezeigt wird. Dies bedeutet lediglich, dass die VC 9.0 Runtime bereits auf Ihrem System installiert ist und dass Sie mit einem 32-Bit-Rechner arbeiten.

Das Setup-Programm beginnt nun mit dem Kopieren der Dateien. Zum Schluss erscheint dann noch eine Erfolgsmeldung.

Visual C++ starten

Visual C++ ist ein Compiler mit einer integrierten Entwicklungsumgebung (kurz IDE für »Integrated Developing Environment«). Für uns als Programmierer be-

deutet dies, dass wir auch bei der Programmierung nicht auf den Komfort einer grafischen Benutzeroberfläche verzichten müssen.

Was ist das?

Bei der Programmerstellung ist der Programmierer auf eine Reihe von Hilfsprogrammen angewiesen (Editor, Compiler, Linker, Debugger), von denen die meisten traditionell Befehlszeilenprogramme sind (Compiler, Linker, Debugger), d.h., sie verfügen über keine Fenster oder grafische Oberflächen und müssen von der Konsole aus (Eingabeaufforderung) ausgeführt werden (vergleiche den weiter unten vorgestellten GNU-Compiler). Die integrierte Entwicklungsumgebung von Visual C++ ist eine Art Über-Programm, von dem aus alle für die Programmentwicklung benötigten Programme aufgerufen werden können. Sie verfügt über einen integrierten Editor, Dialogfelder und Menübefehle zum Aufruf des Compilers und eine ausgereifte Projektverwaltung.

Starten Sie nun die Visual C++-Entwicklungsumgebung, um sich mit ihr vertraut zu machen.

1 Öffnen Sie das Start-Menü und wählen Sie in der Microsoft Visual Studio Express-Programmgruppe den Eintrag für Visual C++ 2010 Express aus.

Nach kurzer Ladezeit erscheint die integrierte Entwicklungsumgebung von Visual C++.

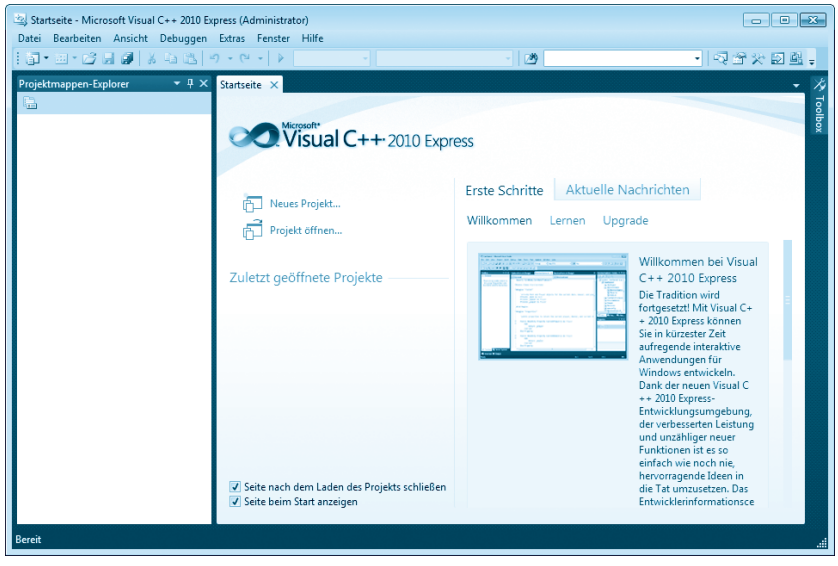

Abbildung 3.3: Die integrierte Entwicklungsumgebung von Visual C++

Neben der Menüleiste und einer Reihe von mehr oder weniger leeren Fensterbereichen gibt es nicht viel zu sehen. Doch das ändert sich, wenn wir mit dem Programmieren beginnen.

Zuvor aber müssen wir noch eine kleine Veränderung vornehmen. Um den Benutzer nicht zu überfordern, hat Microsoft nämlich in seiner unendlichen Fürsorglichkeit die Express Edition werksmäßig so eingestellt, dass nur einer begrenzter Teil der enthaltenen Funktionalität angeboten wird. Dies ist ärgerlich und Sie sollten es sofort ändern.

2 Rufen Sie den Menübefehl *Extras/Einstellungen/Erweiterte Einstellungen* auf.

Ein neues Projekt anlegen

In Visual C++ werden Programme in Form von Projekten verwaltet (die wiederum in Projektmappen organisiert werden).

Was ist das?

Die Quelldateien und Informationen, die zur Erstellung eines Programms benötigt werden, verwaltet der Visual C++-Compiler in Form eines Projekts. Die Projektverwaltung ist für den Programmierer umso wertvoller, je komplexer und umfangreicher die erstellten Programme sind.

Der erste Schritt bei der Programmerstellung mit Visual C++ besteht daher darin, ein passendes Projekt anzulegen.

1 Öffnen Sie das Menü *Datei*, klicken Sie auf den Befehl *Neu/Projekt*.

2 Im Dialogfeld *Neues Projekt* sollte jetzt die Vorlagen-Kategorie *Visual C++* angezeigt werden. Falls nicht, klicken Sie einfach im linken Teilfenster auf den gleichnamigen Link. Wählen Sie dann im mittleren Fenster die Vorlage *Win32-Konsolenanwendung* aus.

Die Auswahl der Projektvorlage ist ein ganz wichtiger Schritt. Durch die Projektvorlage teilen wir Visual C++ mit, was wir programmieren wollen: ein eigenständiges Programm, eine Bibliothek, die von anderen Programmen verwendet werden kann, und so weiter.

Wie Sie sehen, gibt es eine ganze Reihe von Projektvorlagen, d.h., der Visual C++-Compiler bietet uns weit mehr Möglichkeiten, als wir im Moment nutzen können. Macht nichts, wir konzentrieren uns auf unser eigentliches Ziel: die Erstellung eines Programms. Hierfür stehen vier Projektvorlagen zur Verfügung:

- CLR-Konsolenanwendung
- Windows Forms-Anwendung
- Win32-Konsolenanwendung
- Win32-Projekt

Die ersten beiden Projektvorlagen sind für die Erstellung von .NET Framework-Programmen gedacht. Dies sind Programme, die nicht direkt vom Windows-Betriebssystem, sondern von der unter Windows installierten .NET Framework-Laufzeitumgebung ausgeführt werden. Für uns ist das .NET Framework uninteressant, da es vom C++-Programmierer verlangt, dass er sich mit speziellen, nicht standardisierten Syntaxerweiterungen vertraut macht.

Die beiden Win32-Projektvorlagen erzeugen echte »native« C++-Anwendungen. Beide Vorlagen rufen letztlich den *Win32-Anwendungs-Assistenten* auf, der das gewünschte Projekt erstellt – nur dass *Win32-Projekt* den Assistenten vorab für die Erstellung von GUI-Anwendungen konfiguriert, während *Win32-Konsolenanwendung* die Optionen des Assistenten vorab so einstellt, dass eine Konsolenanwendung angelegt wird.

Hinweis

Was Konsolenanwendungen sind und warum wir uns für sie entscheiden, wurde bereits im vorangehenden Kapitel im *Abschnitt* »Von Windows, Fenstern und Konsolen« ausgeführt.

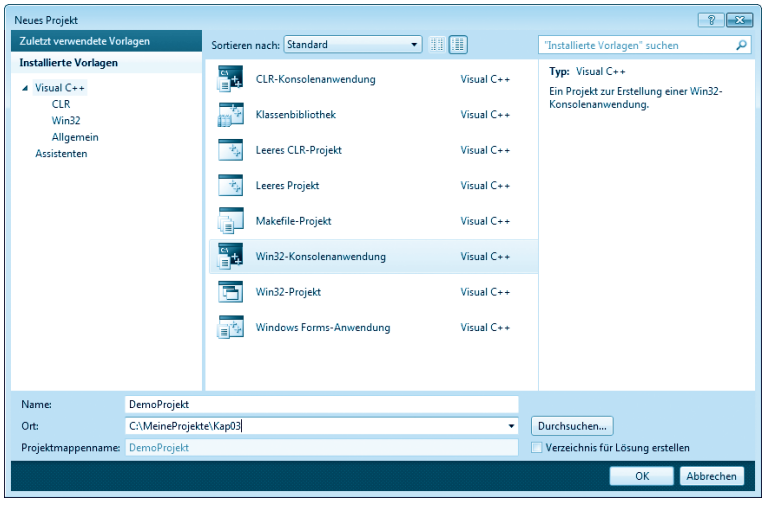

Abbildung 3.4: Neues Projekt anlegen

3 Legen Sie fest, wie das Projektverzeichnis heißen und wo es angelegt werden soll.

Geben Sie Ihrem Projekt unten in dem Eingabefeld einen Namen, beispielsweise *DemoProjekt*. Dieser Name wird auch als Name des Projektverzeichnisses verwendet.

Wählen Sie im Feld *Ort* das übergeordnete Verzeichnis aus, unter dem das Projektverzeichnis eingerichtet werden soll. (Sie können den Verzeichnispfad eintippen oder über das Dialogfeld auswählen, das erscheint, wenn Sie auf die *Durchsuchen*-Schaltfläche klicken.)

Deaktivieren Sie die Option *Verzeichnis für Lösung erstellen*. (Wenn Sie die Option markieren, wird unter dem Speicherort zuerst ein Projektmappenverzeichnis[1] und darunter dann das Projektverzeichnis angelegt, siehe weiter unten den *Abschnitt »*Projektmappen*«*.)

4 Klicken Sie zum Schluss auf *OK*.

Tipp

Erstellen Sie Ihre Projekte unter einem gemeinsamen Pfad – beispielsweise *C:\MeineProjekte* – und legen Sie unter diesem übergeordneten Verzeichnis die Projektverzeichnisse an. Besser noch: Legen Sie für die Beispiele jedes Buchkapitels noch ein eigenes Unterverzeichnis an, wie z.B. *Kapo3*, *Kapo4* etc. So trägt das Projekt aus diesem Kapitel beispielsweise den Namen *DemoProjekt* und die Dateien des Projekts werden in dem Verzeichnis *C:\MeineProjekte\Kapo3* abgespeichert.

Achtung

Wählen Sie möglichst einen Verzeichnispfad und einen Namen ohne Leer- und Sonderzeichen.

1 *Wieso »Projektmappe«? Die Option spricht doch von einem Verzeichnis für die »Lösung«? Nun, hierbei handelt es sich um einen Übersetzungsfehler. Im Englischen heißt es »solution«, was wörtlich übersetzt zwar »Lösung« bedeutet, tatsächlich aber der englische Begriff für die Projektmappe ist.*

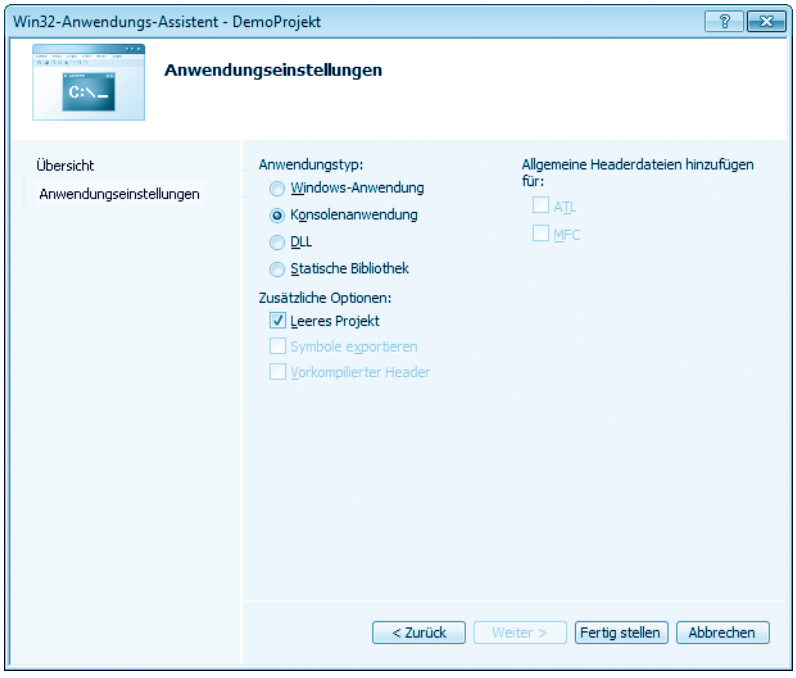

Abbildung 3.5: Die Projektvorlage konfigurieren

5 Deaktivieren Sie auf der zweiten Seite des Assistenten zuerst die Option *Vorkompilierter Header*. Aktivieren Sie dann die Option *Leeres Projekt* und kontrollieren Sie, ob weiter oben *Konsolenanwendung* ausgewählt ist. Klicken Sie dann auf *Fertig stellen*.

Vorkompilierte Header dienen dazu, die Programmerstellung zu beschleunigen. Sie werden bei der ersten Kompilierung erstellt und können nachfolgende Kompilierungen beschleunigen. Wenn Sie an größeren Projekten arbeiten, ist dies eine recht nützliche Option. Für unsere kleinen Beispielprogramme können wir allerdings auf den »Header«, der viel Speicherplatz belegt, verzichten.

Wenn Sie die Option *Leeres Projekt* nicht aktivieren, legt Visual C++ für Sie eine CPP-Quelltextdatei mit einem einfachen Programmgerüst an. Wir verzichten allerdings auf dieses Programmgerüst, da es a) nur wenig Arbeitserleichterung bringt und b) kein standardisiertes C++ verwendet.

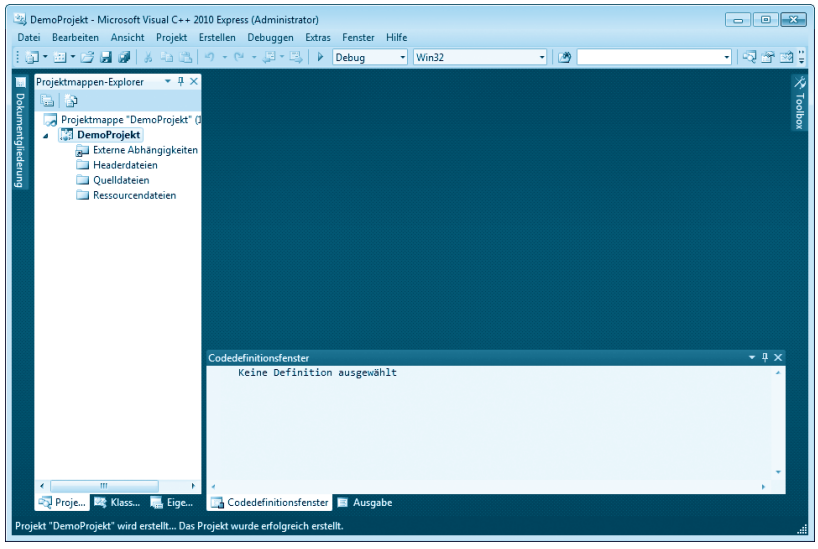

Abbildung 3.6: Die IDE nach dem Anlegen des Projekts

In der IDE hat sich jetzt etwas getan. In der Titelleiste kann man lesen, welches Projekt gerade bearbeitet wird, und im Projektmappen-Explorer (dies ist das Fenster links) kann man sich über den Aufbau des Projekts informieren. Im Moment gibt es dort aber noch nicht viel zu sehen, weil unser Projekt noch leer ist.

Projektmappen

Visual C++ organisiert Projekte in Projektmappen. Für fortgeschrittene Programmierer ist dies interessant, insofern sie so ihre Projekte in Gruppen zusammenfassen können – beispielsweise ein Programm und eine für das Programm geschriebene Bibliothek oder die verschiedenen Versionen eines Programms. Für uns sind die Projektmappen weniger interessant und wir vereinfachen uns die Sache, indem wir sie einfach ignorieren. Sie werden deswegen zwar trotzdem angelegt, so dass jedes unserer Projekte in seiner eigenen Projektmappe residiert, aber dies stört ja nicht.

Wenn Sie trotzdem mit Projektmappen arbeiten möchten oder einfach aus Neugier einmal mehrere Projekte in einer gemeinsamen Projektmappe organisieren möchten, habe ich einige Hinweise für Sie:

- Achten Sie darauf, dass Sie beim Anlegen des ersten Projekts der Projektmappe ein Projektmappenverzeichnis anlegen (Option *Verzeichnis für Lösung erstellen* aktivieren).

- Um der Projektmappe ein weiteres Projekt hinzuzufügen, rufen Sie wie gewohnt den Befehl *Datei/Neu/Projekt* auf, wählen dann aber in dem aufklappbaren Listenfeld unten im Dialogfeld die Option *Hinzufügen* aus (Voreinstellung ist *Neue Projektmappe erstellen*).
- Projektmappen-Dateien haben die Dateierweiterung *.sln*. Um eine komplette Projektmappe zu öffnen, rufen Sie den Befehl *Datei/Öffnen/Projekt/Projektmappe* auf und wählen die gewünschte SLN-Datei aus. (Oder doppelklicken Sie einfach im Windows Explorer auf die SLN-Datei.)
- In Visual C++ können Sie nicht nur die Quelltexte Ihrer Programme bearbeiten, sondern die Programme auch gleich kompilieren und ausführen. Gibt es allerdings in einer Projektmappe mehrere Projekte, müssen Sie Visual C++ mitteilen, welches Projekt kompiliert und ausgeführt werden soll. Rufen Sie dazu den Menübefehl *Projekt/Als Startprojekt festlegen* auf.

Den Quelltext aufsetzen

Bevor wir den Quelltext zu unserem Programm eingeben können, müssen wir eine Quelldatei in unser Projekt aufnehmen.

Quelltextdateien anlegen

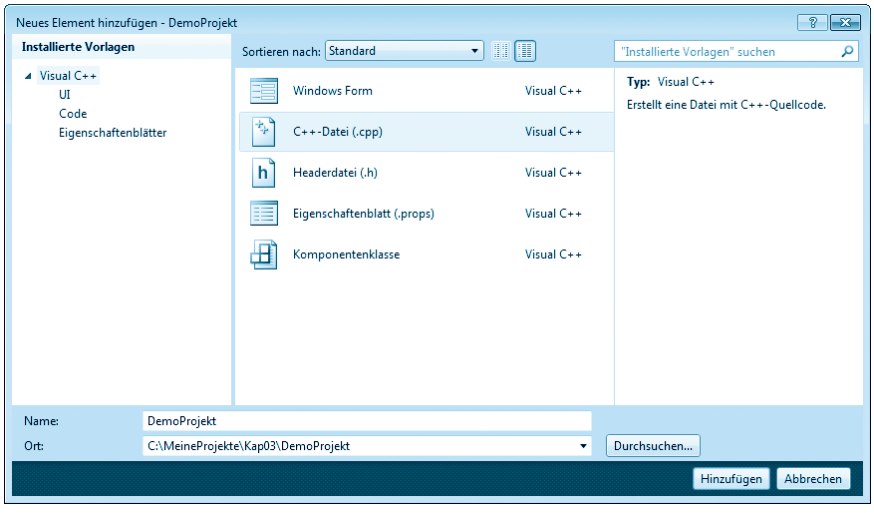

Abbildung 3.7: Dem Projekt eine Quelltextdatei hinzufügen

1 Klicken Sie mit der rechten Maustaste im Projektmappen-Explorer auf den Projektknoten (in unserem Beispielprojekt ist dies der Knoten mit dem fett dargestellten Namen *DemoProjekt*) und rufen Sie im Kontextmenü den Befehl *Hinzufügen/Neues Element* auf

2 Wählen Sie im erscheinenden Dialogfeld die Vorlage *C++-Datei (.cpp)* aus, gehen Sie einen Namen für die Datei an und klicken Sie auf *Hinzufügen*.

> **Hinweis**
>
> Unsere Beispielprogramme bestehen meist nur aus einer Quelltextdatei, die dann der Einfachheit halber und zur leichteren Identifizierung den Namen des Projekts trägt.

Die Quelldatei wird automatisch angelegt und in den integrierten Editor geladen.

Die Dateien eines Projekts

Standardmäßig sehen Sie im linken Teil der Entwicklungsumgebung von Visual C++ den Projektmappen-Explorer. (Falls nicht, können Sie ihn über den Menübefehl *Ansicht/Projektmappen-Explorer* aufrufen.) Im Projektmappen-Explorer können Sie sich darüber informieren, welche Dateien zu Ihrem Projekt gehören, und diese bei Bedarf per Doppelklick in den Editor laden.

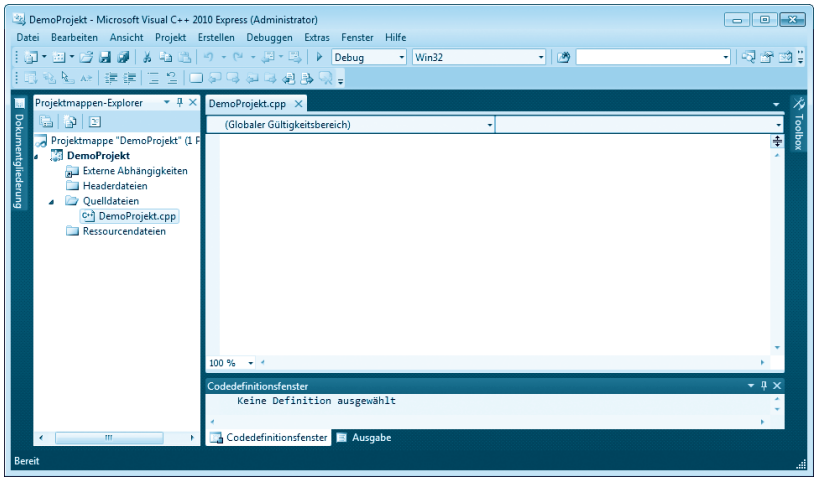

Abbildung 3.8: Projektmappen-Explorer und geladene Quelldatei in der IDE

> **Tipp**
>
> Durch Doppelklick auf den Knoten einer Quelldatei können Sie diese zur Bearbeitung in den Editor laden.

Den Programmquelltext eingeben

Nachdem die Datei als Teil des Projekts angelegt und in den Editor geladen wurde, können Sie den Quelltext des Programms eingeben.

> **Hinweis**
>
> Wenn Sie nicht sicher sind, ob die richtige Datei in den Editor geladen wurde, schauen Sie einfach in die Registerleiste über dem Editorfenster. In den Reitern stehen die Namen der Quelldateien.

3 Tippen Sie den folgenden Programmquelltext ein.

```
// Hallo Welt-Programm

#include <iostream>
using namespace std;

int main()
{
   cout << "Hallo Welt!" << endl;

   return 0;
}
```

Tippen Sie den Quelltext bitte genauso ab, wie er oben aufgelistet ist. Was dieser Code im Einzelnen bedeutet, werden Sie im nächsten Kapitel erfahren. Wenn Sie unsicher sind, ob Sie den Quelltext richtig abgetippt haben, kopieren Sie den Quelltext einfach aus der Datei *DemoProjekt.cpp* auf der Buch-CD.

> **Achtung**
>
> In C++ wird zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie also beispielsweise `Main()` statt `main()` eintippen, ist dies ein Fehler!

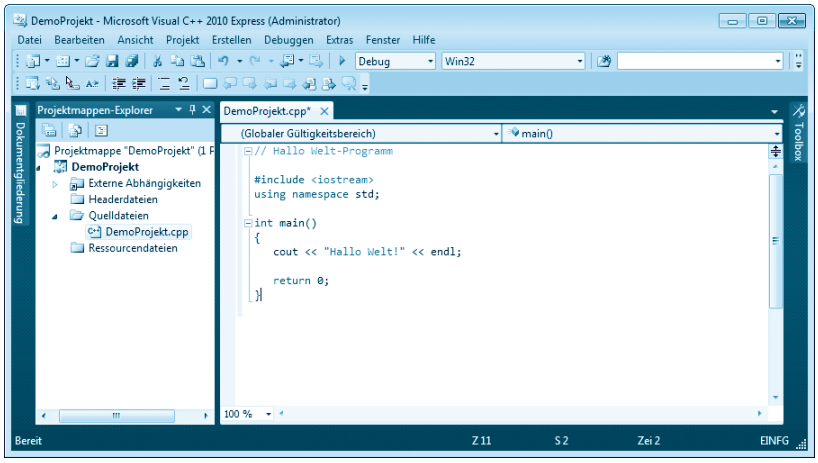

Abbildung 3.9: Der Programmcode wurde in die Datei DemoProjekt.cpp eingetippt.

Speichern und erstellen

Bevor wir das Programm kompilieren, speichern wir den Quellcode ab. Dies ist zwar nicht unbedingt notwendig, gibt uns aber das sichere Gefühl, dass unser Programmcode nicht verloren geht, wenn die Visual C++-Entwicklungsumgebung beim Kompilieren (oder späteren Ausführen) des Programms abstürzen sollte.

1 Speichern Sie die Dateien des Projekts durch Aufruf des Befehls *Datei/... Speichern* oder durch Drücken der Tastenkombination Strg + S.

> **Hinweis**
>
> Wenn die aktuell geladene Quelldatei Änderungen enthält, die noch nicht abgespeichert wurden, sehen Sie in der Titelleiste ein Sternchen neben dem Dateinamen.

Erstellen

Bis jetzt haben wir nicht mehr als eine ganz normale Textdatei, deren Inhalt zufälligerweise der C++-Sprachspezifikation entspricht. Das mag nicht sonderlich aufregend klingen, aber unter Umständen bedeutet es, dass wir nur noch fünf Minuten von unserem ersten eigenen Programm entfernt sind.

1 Rufen Sie den Befehl *Erstellen/Projektmappe erstellen* auf, um den Quelltext zu kompilieren und die EXE-Datei erstellen zu lassen.

Im unteren Bereich der Visual C++-IDE wird jetzt das Ausgabefenster eingeblendet, in dem Sie den Fortschritt der Erstellung verfolgen können.

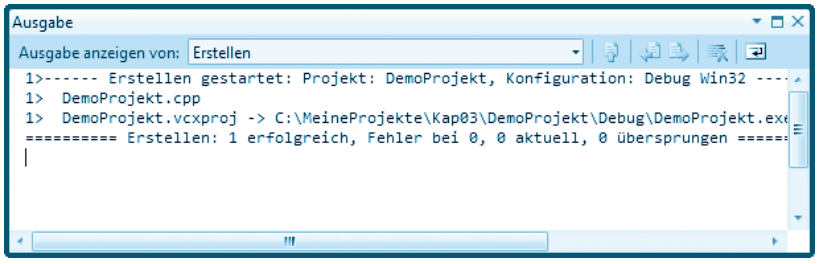

Abbildung 3.10: Null Fehler – Super!!!

Nur selten kommt es vor, dass man beim Aufsetzen des Quelltextes keinen Fehler macht. Oft sind es ganz unnötige Tippfehler, die dem C++-Novizen das Studium zur Hölle machen. Dabei sind diese so genannten »syntaktischen Fehler« noch recht harmlos, denn sie werden – im Gegensatz zu den logischen Fehlern – vom Compiler entdeckt und meist recht gut lokalisiert.

Syntaxfehler beheben

Syntaktische Fehler (im Gegensatz zu logischen Fehlern, die sich dadurch bemerkbar machen, dass das Programm nicht das tut, wofür es geschrieben wurde) werden bereits vom Compiler entdeckt und angezeigt.

1 Bauen Sie in den Quelltext einen Fehler ein. Löschen Sie dazu beispielsweise eine der spitzen Klammern aus der `cout`-Zeile.

```
cout << "Hallo Welt!" << endl;
```

2 Rufen Sie erneut den Befehl *Erstellen/Projektmappe erstellen* auf oder drücken Sie die Taste F7.

Im Ausgabefenster erscheinen nun einige Fehlermeldungen. Die Fehlermeldungen des *Ausgabe*-Fensters sind allerdings recht unübersichtlich. Rufen Sie daher die *Fehlerliste* auf (Menübefehl *Ansicht/Weitere Fenster/Fehlerliste*) und klicken Sie, falls in der Fehlerliste nichts zu sehen ist, auf den Reiter *Fehler* (vgl. Abbildung 3.11).

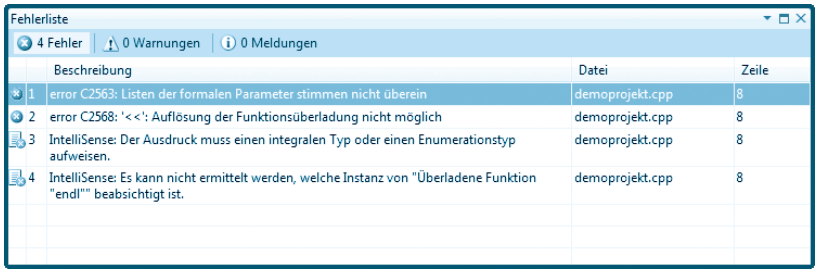

Abbildung 3.11: Am übersichtlichsten werden die Fehlermeldungen des Compilers in der Fehlerliste angezeigt.

3 Doppelklicken Sie auf die erste Fehlermeldung.

Im Editor erscheint am linken Rand ein Pfeil, der die Zeile markiert, in der der Compiler den Fehler vermutet. Suchen Sie in der Zeile nach dem Fehler. Eventuell kann Ihnen der Text der Fehlermeldung dabei behilflich sein.

4 Korrigieren Sie den Fehler.

```
cout << "Hallo Welt!" << endl;
```

5 Rufen Sie den Befehl *Erstellen/Projektmappe erstellen* auf, um das Programm erneut kompilieren und linken zu lassen.

> **Tipp**
>
> Arbeiten Sie die Fehlerliste immer von oben nach unten ab und erstellen Sie nach jedem korrigierten Fehler neu. Manchmal handelt es sich bei den unteren Meldungen nämlich um Folgefehler, die nach der Kompilation automatisch verschwinden.

Das Programm testen

Wenn der Compiler das Programm ohne Probleme erstellen konnte, bedeutet dies noch nicht, dass das Programm auch korrekt arbeitet. Der letzte Schritt bei der Programmerstellung besteht daher darin, das Programm auszuführen und zu testen, ob es das macht, wofür es programmiert wurde.

1 Rufen Sie den Befehl *Debuggen/Starten ohne Debugging* auf.

Die IDE lädt das Programm und führt es in einem eigenen Konsolenfenster aus.

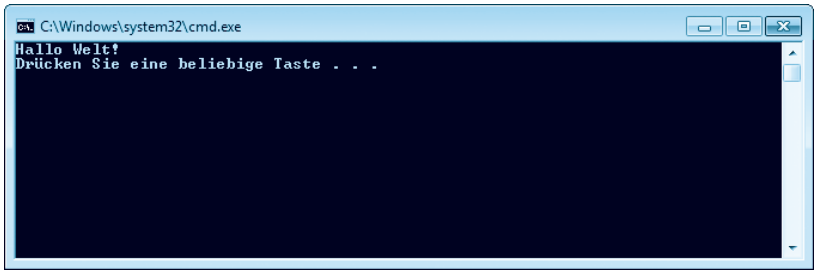

Abbildung 3.12: Die Ausgabe des Programms

Im Falle unseres Beispielprogramms gibt es nicht viel zu testen. Sie müssen lediglich schauen, ob die Ausgabe korrekt ist.

> **Hinweis**
>
> Die Aufforderung »Drücken Sie eine beliebige Taste« ist übrigens nicht Teil unseres Programms. Sie wird von Visual C++ erzeugt und verhindert, dass das Konsolenfenster automatisch geschlossen wird, nachdem das Programm beendet wurde – so haben wir Zeit, uns die Ausgaben des Programms in Ruhe anzuschauen.

2 Drücken Sie eine beliebige Taste, um das Konsolenfenster zu schließen.

Das Programm ausführen

Natürlich will man nicht immer Visual C++ aufrufen, um ein selbst erstelltes Programm auszuführen. Aber das ist ja auch nicht nötig. Die EXE-Datei des Programms ist auf der Festplatte im Verzeichnis des Projekts abgespeichert (Unterverzeichnis *Debug* oder *Release*) und kann direkt aufgerufen werden.

> **Hinweis**
>
> Erschrecken Sie nicht wegen der vielen Dateien im *Debug*-Verzeichnis Ihres Projekts (*DemoProjekt.ilk*, *DemoProjekt.pdb* etc.). Dies sind lediglich Hilfsdateien, die das Kompilieren und Linken beschleunigen. Wenn das Programm fertig und ausgetestet ist, können Sie die EXE-Datei z.B. in das Projektverzeichnis hochkopieren und das gesamte *Debug*-Verzeichnis danach löschen.

1 Öffnen Sie ein Konsolenfenster.

Unter Windows 7/Vista/XP öffnen Sie dazu das Start-Menü und wählen unter *Alle Programme\Zubehör* den Eintrag *Eingabeaufforderung*. Unter älteren Windows-Betriebssystemen heißt die Konsole manchmal auch *MS-DOS-Eingabeaufforderung* oder ist direkt unter *Start/Programme* zu finden.

2 Wechseln Sie in der Konsole mithilfe des *cd*-Befehls in das Verzeichnis mit der EXE -Datei, also beispielsweise:

```
Prompt> cd C:\MeineProjekte\Kap03\DemoProjekt\Debug
```

3 Starten Sie die Anwendung über ihren Namen.

```
Prompt> DemoProjekt
```

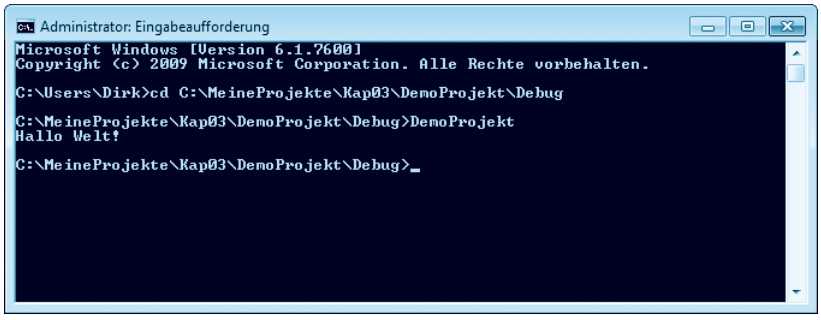

Abbildung 3.13: Ausführung des Programms von der Konsole aus

Hinweis

Mithilfe des Befehls `dir` oder `dir /p` können Sie sich den Inhalt des aktuellen Verzeichnisses anzeigen lassen. Weitere Informationen zur Windows-Eingabeaufforderung finden Sie in dem Tutorial »Arbeiten mit der Windows-Konsole«, das Sie von der Website `www.carpelibrum.de` herunterladen können.

Hinweis

Wie Sie die Programme per Doppelklick im Windows Explorer oder über eine Desktop-Verknüpfung starten, lesen Sie weiter unten im *Abschnitt* »Noch einmal: Ausführen von Konsolenprogrammen«.

Visual C++ beenden

Nach getaner Arbeit wird aufgeräumt. Zuerst schließen wir das Projekt, genauer gesagt, die Projektmappe unseres Projekts.

Projektmappen schließen

Wenn Sie mit der Arbeit an Ihrem Programm fertig sind oder ein neues Projekt beginnen wollen, schließen Sie die Projektmappe des aktuellen Projekts.

1 Rufen Sie den Befehl *Datei/Projektmappe schließen* auf.

Visual C++ beenden

Wenn Sie Ihre Arbeit mit Visual C++ ganz beenden wollen...

1 ... rufen Sie den Befehl *Datei/Beenden* auf.

Bestehende Projekte neu öffnen

Wenn Sie am nächsten Tag an Ihrem Projekt weiterarbeiten wollen, rufen Sie zuerst Visual C++ und laden dann das gewünschte Projekt.

Es gibt mehrere Möglichkeiten, um ein bestehendes Programm wieder zu öffnen:

- Sie können den Befehl *Datei/Öffnen/Projekt/Projektmappe* aufrufen (und die VCXPROJ-Projektdatei aus dem Projektverzeichnis laden. (So können Sie auch die Beispielprojekte auf der Buch-DD öffnen.)
- Die zuletzt bearbeiteten Projektmappen können Sie über den Befehl *Datei/Zuletzt geöffnete Projekte und Projektmappen* geladen werden.
- Sie können die IDE so einstellen, dass beim Start von Visual C++ automatisch die zuletzt bearbeitete Projektmappe geladen wird.

Letztes Projekt automatisch beim Start laden

Wenn Sie möchten, können Sie Visual C++ so einrichten, dass beim Start von Visual C++ automatisch die Projektmappe des zuletzt bearbeiteten Programms geladen wird.

1 Rufen Sie den Befehl *Extras/Optionen* auf und wählen Sie links die Kategorie *Umgebung/Start* aus.

2 Wählen Sie im Listenfeld *Beim Start* die Option *Letzte Projektmappe laden* aus.

3 Klicken Sie auf *OK*.

Programmerstellung mit dem g++-GNU-Compiler (Linux)

Als Beispiel für einen Linux-Compiler sei hier die Programmerstellung mit dem GNU C++-Compiler, im Folgenden kurz g++ genannt, beschrieben.

Installation

Auf vielen Systemen ist der g++-Compiler standardmäßig installiert. Sie können dies testen, indem Sie ein Konsolenfenster öffnen und den Befehl g++ abschicken. Erscheint eine Meldung des Compilers (beispielsweise ein Hinweis auf eine fehlende Dateiangabe), ist alles bestens. Erscheint ein Befehl, dass der Compiler nicht gefunden werden kann, müssen Sie den Compiler nachinstallieren. Ein Besuch bei `gcc.gnu.org` ist dabei nur selten nötig, meist ist der GNU-C++-Compiler g++ im Umfang der Linux-Distribution enthalten und kann von der Linux-Installations-DVD nachinstalliert werden.

Programme erstellen und kompilieren

Abbildung 3.14: Das Konsolenfenster

1 Öffnen Sie ein Konsolenfenster.

Wie Ihr Konsolenfenster aussieht und mit welchem Befehl es aufgerufen wird, hängt von Ihrer Linux-Version und dem verwendeten Window-Manager ab. Beachten Sie aber, dass die Konsole unter Linux oft auch als »Terminal« bezeichnet wird und nicht selten mit einem schwarzen Fenster als Programmsymbol gekennzeichnet ist.

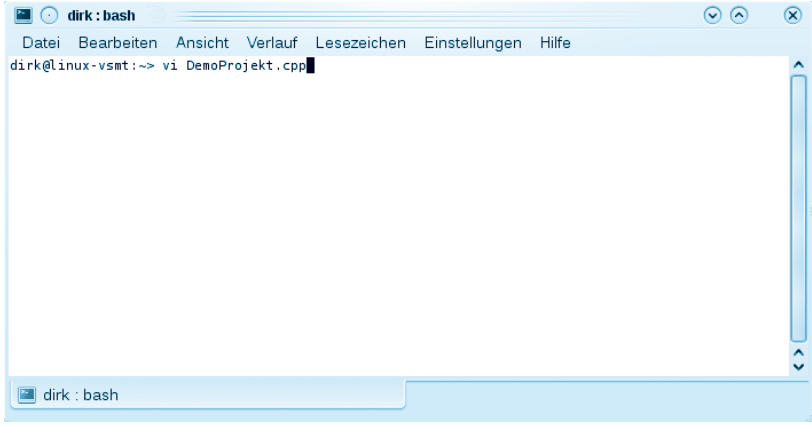

Abbildung 3.15: Aufruf des vi

2 Legen Sie mit dem Editor *vi* eine neue Quelltextdatei an.

Unter Linux/KDE des *vi* können Sie auch jeden beliebigen anderen Text-Editor verwenden, beispielsweise den *emacs*, *KEdit* oder *KWrite* unter KDE.

Wenn Sie den *vi* verwenden, geben Sie im Aufruf gleich den Namen der neu anzulegenden Quelldatei an. Nach dem Aufruf können Sie den Text der neuen Datei direkt im Konsolenfenster (in dem jetzt der *vi* ausgeführt wird) aufsetzen.

3 Drücken Sie die i-Taste, um in den Einfügemodus zu wechseln.

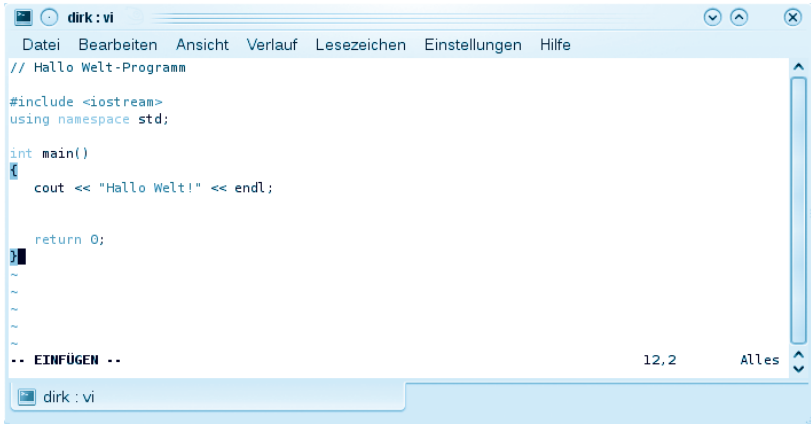

Abbildung 3.16: Quelltext eintippen

4 Geben Sie den Programmquelltext ein. (Wenn Sie den Quelltext in der Abbildung nicht lesen können, blättern Sie zurück zum Abschnitt »Programmerstellung mit Visual C++ (Windows)«, wo der Quelltext noch einmal abgedruckt ist.)

> **Achtung**
>
> In C++ wird zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie also beispielsweise `Main()` statt `main()` eintippen, ist dies ein Fehler!

5 Drücken Sie die Esc-Taste, um in den Befehlsmodus zu wechseln.

Abbildung 3.17: Speichern und beenden

6 Tippen Sie `:wq` ein, um die Datei zu speichern und den *vi* zu beenden.

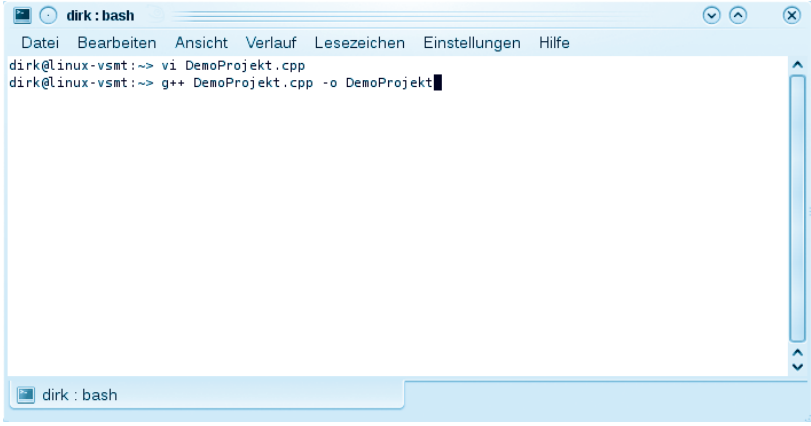

Abbildung 3.18: Kompilieren mit dem g++-Compiler

7 Rufen Sie von der Konsole aus den g++-Compiler auf.

Übergeben Sie dem g++-Compiler in der Kommandozeile den Namen der zu kompilierenden Datei sowie den Schalter `-o` mit dem gewünschten Namen für die ausführbare Datei.

Abbildung 3.19: Ausführung in einem Konsolenfenster

8 Führen Sie das Programm aus.

Noch einmal: Ausführen von Konsolenprogrammen

Was passiert eigentlich, wenn man versucht, ein bereits erstelltes Konsolenprogramm durch Doppelklick aus einem Ordnerfenster (Windows Explorer etc.) heraus zu starten?

In diesem Fall öffnet das Betriebssystem für das Programm, das selbst ja über kein eigenes Fenster verfügt, ein Konsolenfenster. Leider schließt das Betriebssystem das Konsolenfenster auch direkt wieder, wenn das Programm beendet ist. Im Falle unseres Test-Programms bliebe Ihnen dadurch kaum Zeit, die Ausgabe des Programms zu kontrollieren, ja womöglich werden Sie das Öffnen und Schließen des Konsolenfensters nur als kurzes Aufblinken wahrnehmen!

In diesem Fall können Sie auf zweierlei Wegen Abhilfe schaffen.

Erstens: Sie fügen am Ende des Programmquelltextes den Methodenaufruf `cin.get()` ein und erstellen das Programm neu.

```
// Hallo Welt-Programm

#include <iostream>
using namespace std;

int main()
{
   cout << "Hallo Welt!" << endl;

   cin.get();
   return 0;
}
```

Die Methode `cin.get()` wartet darauf, dass der Anwender über die Tastatur ein Zeichen eingibt und mit der ↵-Taste abschickt. Hier wird sie ein wenig zweckentfremdet. Wir nutzen sie dazu, die Beendigung des Programms so lange hinauszuzögern, bis der Anwender – in diesem Falle wir – die ↵-Taste drückt.

Die zweite Möglichkeit ist, selbst ein Konsolenfenster zu öffnen und das Programm in diesem ausführen zu lassen – so wie wir es zum Abschluss der Programmerstellung getan haben.

Rätselhaftes C++

Wer sind wir? Woher kommen wir? Wohin gehen wir? Nichts steht isoliert in der Welt. Auch nicht die Programmiersprache C++. Eine der Ursprünge von C++ haben Sie bereits kennen gelernt: die Sprache C. Doch welche Programmiersprache stand für die objektorientierten Konzepte Modell? Und in welche Richtung wird sich C++ entwickeln?

Lösung: Bei der Entwicklung der objektorientierten Konzepte wurde Stroustrup von der objektorientierten Programmiersprache Simula67 inspiriert. Einen direkten Nachfolger von C++ gibt es nicht und wird es wohl auch nie geben. Es gibt jedoch zwei Programmiersprachen, die stark von C++ inspiriert sind: Java und C#. Beide, Java wie C#, sind rein objektorientiert, d.h., sie zwingen den Programmierer, seine Quelltexte von Anfang an aus Klassendefinitionen aufzubauen. Außerdem werden Java- und C#-Programme interpretiert, d.h., auf dem System, auf dem ein Java- (C#-) Programm ausgeführt werden soll, muss eine entsprechende Laufzeitumgebung (für Java die JRE, für C# das .NET-Framework) installiert sein, die den Programmcode ad hoc in Maschinencode umwandelt und ausführen lässt.

Copyright

Daten, Texte, Design und Grafiken dieses eBooks, sowie die eventuell angebotenen eBook-Zusatzdaten sind urheberrechtlich geschützt. Dieses eBook stellen wir lediglich als **persönliche Einzelplatz-Lizenz** zur Verfügung!

Jede andere Verwendung dieses eBooks oder zugehöriger Materialien und Informationen, einschließlich

- der Reproduktion,
- der Weitergabe,
- des Weitervertriebs,
- der Platzierung im Internet, in Intranets, in Extranets,
- der Veränderung,
- des Weiterverkaufs und
- der Veröffentlichung

bedarf der **schriftlichen Genehmigung** des Verlags. Insbesondere ist die Entfernung oder Änderung des vom Verlag vergebenen Passwortschutzes ausdrücklich untersagt!
Bei Fragen zu diesem Thema wenden Sie sich bitte an: info@pearson.de

Zusatzdaten

Möglicherweise liegt dem gedruckten Buch eine CD-ROM mit Zusatzdaten bei. Die Zurverfügungstellung dieser Daten auf unseren Websites ist eine freiwillige Leistung des Verlags. **Der Rechtsweg ist ausgeschlossen.**

Hinweis

Dieses und viele weitere eBooks können Sie rund um die Uhr und legal auf unserer Website herunterladen:

http://ebooks.pearson.de