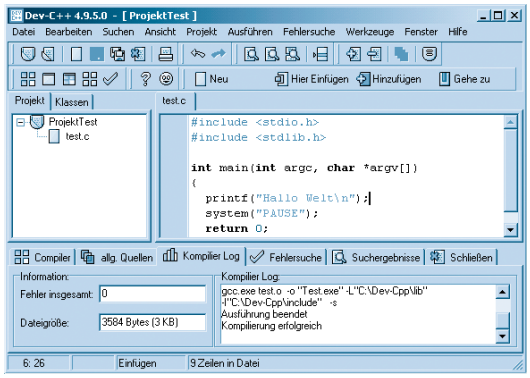
Kapitel 3

Wie man eigene Programme erstellt



In diesem Kapitel geht es darum, sich mit einem Compiler vertraut zu machen. Dabei erfahren Sie, wie Sie eigene Programme auf den Systemen Windows und Linux erstellen. Unter Windows werden Sie den kostenlos erhältlichen Bloodshed Dev-C++-Compiler kennen lernen, der eine eigene Entwicklungsumgebung beinhaltet. Für Linux-Anwender gibt es zum GNU-C-Compiler eine kurze Einführung.

Bitte lesen Sie dieses Kapitel aufmerksam durch, denn die hier beschriebenen Techniken benötigen Sie, um die in den nachfolgenden Kapiteln beschriebenen Programme eingeben, ändern und starten zu können.

Das können Sie schon:

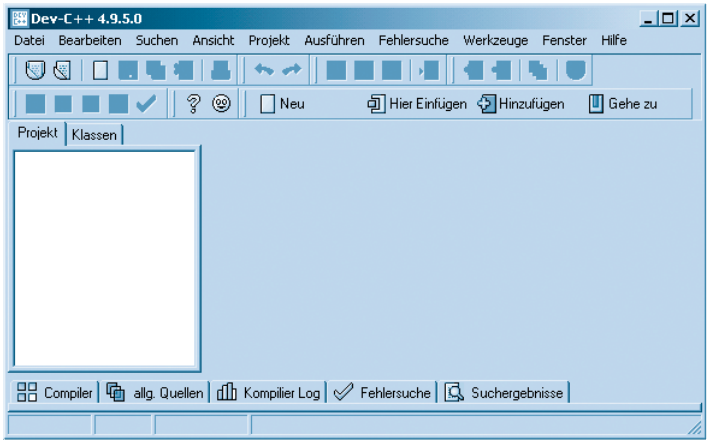
Das lernen Sie neu:

Verwendung des Bloodshed Dev-C++-Compilers

Der *Bloodshed Dev-C++* ist ein kostenloser Compiler für Windows mit einer leistungsfähigen Entwicklungsumgebung, der sich nicht vor den kommerziellen Produkten verstecken muss. Mit Einführung der Version 5 gibt es jetzt endlich auch eine deutschsprachige Benutzeroberfläche.

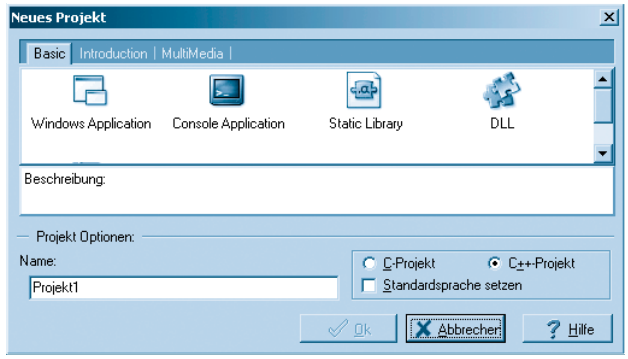
Sie können den Bloodshed Dev-C++-Compiler unter der Webadresse `http://www.bloodshed.net/dev/index.html` herunterladen.

Es wird davon ausgegangen, dass Sie den Compiler bereits heruntergeladen und installiert haben. Starten Sie den Compiler über das START-Menü. Daraufhin wird die Entwicklungsumgebung gestartet.

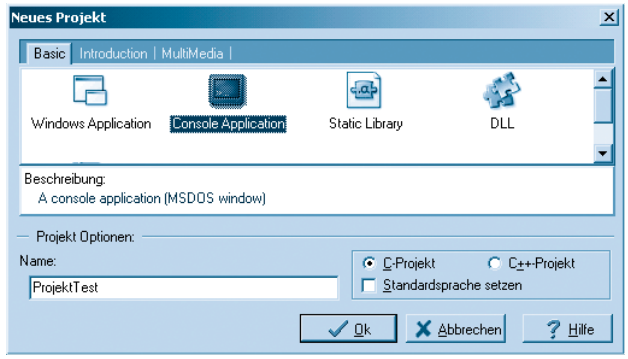


In den folgenden Schritten erfahren Sie, wie Sie ein neues Projekt anlegen, einen Quellcode eingeben und aus diesem eine ausführbare Datei erzeugen.

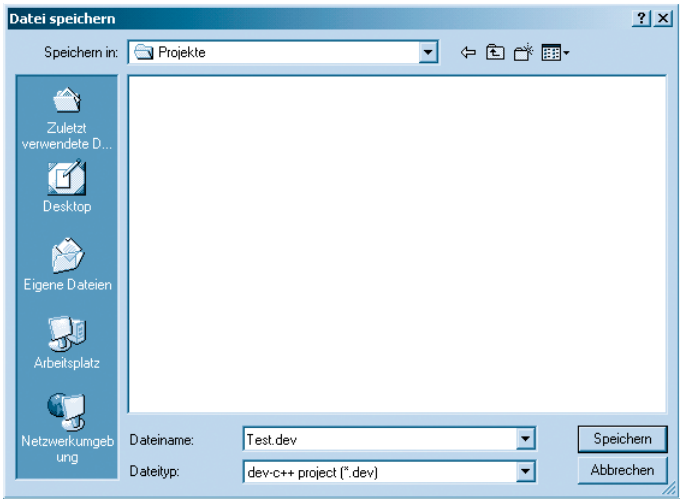
1 Wählen Sie DATEI/NEU/PROJEKT, woraufhin das Dialogfeld NEUES PROJEKT mit dem Register BASIC angezeigt wird.



2 Klicken Sie das Symbol *Console Application* an. Tragen Sie im Eingabefeld NAME den Projektnamen (im Beispiel *ProjektTest*) ein und wählen Sie dann die Option C-PROJEKT. Daraufhin wird die OK-Schaltfläche aktiviert. Klicken Sie auf diese.



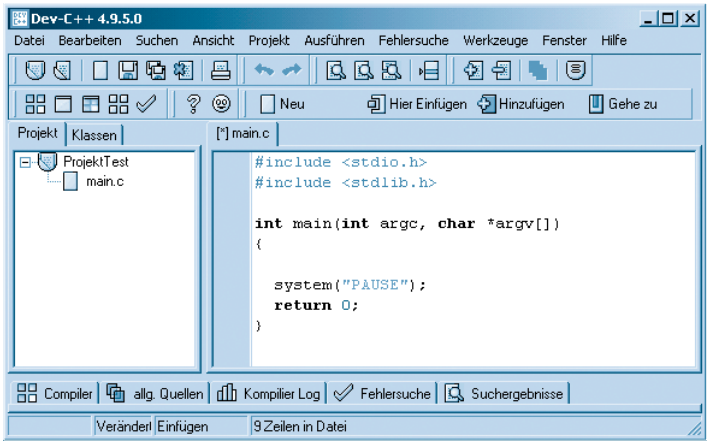
3 Speichern Sie das gerade angelegte Projekt mit der Erweiterung .dev in einem Verzeichnis Ihrer Wahl.



In der Entwicklungsumgebung sehen Sie jetzt ein Programmgrundgerüst mit dem Namen *main.c*. An dem Sternchen (*) neben MAIN.C können Sie erkennen, dass der Programmquellcode noch nicht gespeichert wurde.

Tipp

Dieses Grundgerüst befindet sich im Verzeichnis ...\Templates unter dem Namen ConsoleApp_c.txt. (Das Unterverzeichnis Templates finden Sie in dem Verzeichnis, in dem Sie den Bloodshed Dev-C++-Compiler installiert haben; standardmäßig lautet es C:\Dev-Cpp.)



Hinweis

Die Zeile `system("PAUSE");` *sorgt dafür, dass das Konsolenfenster beim Ausführen und dem anschließenden Beenden des Programms nicht sofort wieder geschlossen wird. Die Zeile* `#include <stdlib.h>` *benötigt man für die Funktion* `system()`*; sie ist eigentlich nicht Bestandteil des Hallo Welt-Grundgerüsts.*

4 Fügen Sie dem Quellcode folgende Zeile hinzu. Vergessen Sie dabei das Semikolon (;) am Zeilenende nicht (in C müssen Anweisungen mit einem Semikolon abgeschlossen werden):

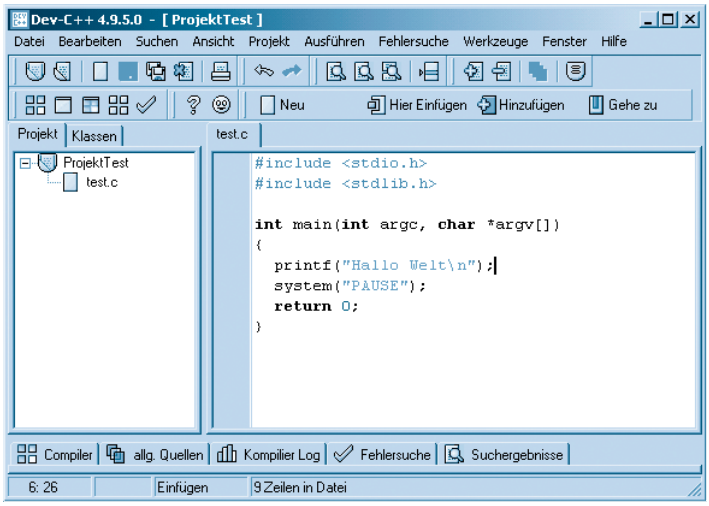
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  printf("Hallo Welt\n");
  system("PAUSE");
  return 0;
}
```

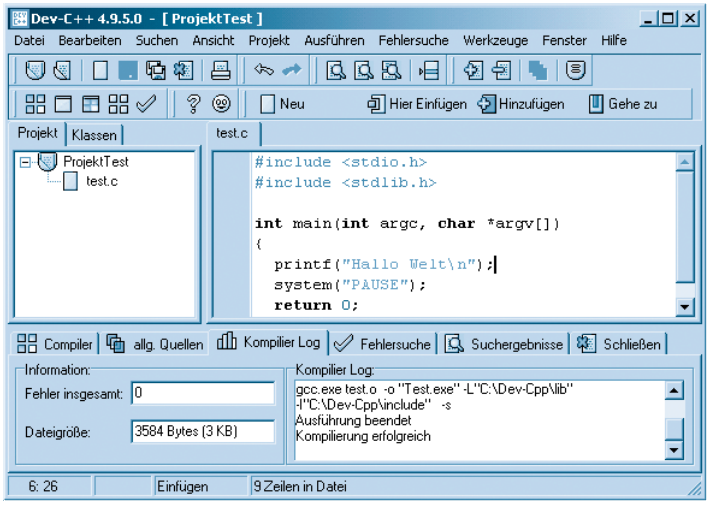
Hinweis

Die Parameter `int argc` *und* `char *argv[]` *zwischen den Klammern der* `main()`*-Funktion können Sie auch entfernen; sie werden in diesem Buch nicht benötigt. (Diese Klammern dienen dazu, ein Programm mit Argumenten aus der Konsole aufzurufen, wodurch einem Programm besondere Werte übergeben werden können, mit denen zum Beispiel das Verhalten des Programms beeinflusst werden kann.)*

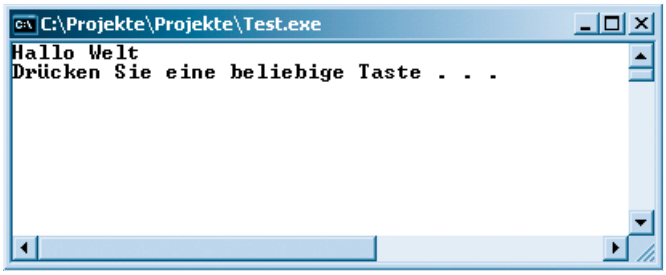
5 Speichern Sie den Programmquellcode unter dem Namen *test.c*.



6 Wählen Sie AUSFÜHREN/KOMPILIEREN. Im unteren Bereich der Entwicklungsumgebung können Sie den Fortschritt der Kompilierung beobachten. Wenn Sie keinen Fehler gemacht haben, sieht alles folgendermaßen aus:



7 Starten Sie das Programm mit AUSFÜHREN/AUSFÜHREN.



Die Ausgabe `Drücken Sie eine beliebige Taste...` hat ihre Ursache im Funktionsaufruf `system("PAUSE")`.

> **Tipp**
>
> *Wenn Sie ein Programm kompilieren und danach sofort ausführen möchten, müssen Sie nicht jedes Mal zwei Kommandos hintereinander anwählen. Drücken Sie stattdessen einfach* F9.

Ausführen von Programmen

Sie müssen ein selbst entwickeltes Programm aber nicht immer aus der Entwicklungsumgebung aufrufen und starten. Sie können ein Programm ebenso starten wie jedes andere Windows-Programm auch, indem Sie zum Beispiel im Windows-Explorer auf die entsprechende exe-Datei doppelklicken. Bei Konsolenprogrammen sieht man allerdings meist das Programm nur kurz aufflackern, da Windows Konsolenprogramme wieder schließt, sobald sie sich beenden. Zeilen wie `system("PAUSE");` baut man aber in der Regel nicht in ein fertiges Programm mit ein.

Daher ist die beste Lösung, ein Konsolenprogramm auszuführen, die (MS-DOS-)Eingabeaufforderung; für diese ist ein Konsolenprogramm auch konzipiert.

Hinweis

Sollten Sie vorhaben, die Programme immer aus der Konsole aufzurufen, sollten Sie folgende zwei Zeilen aus dem Programm entfernen:

```
#include <stdlib.h>
…
system("PAUSE");
```

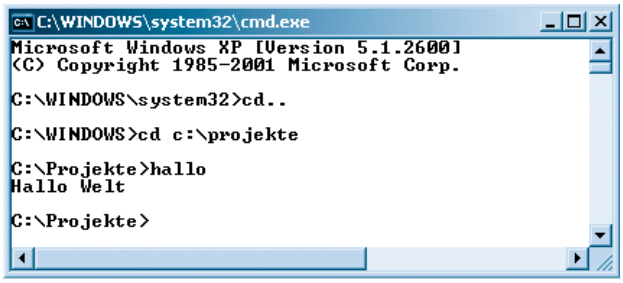
Sie ersparen sich damit den Tastendruck nach jedem Beenden eines Konsolenprogramms.

Wenn Sie möchten, können Sie die Datei ConsoleApp_c.txt im Verzeichnis ...\Templates anpassen (das heißt dort diese beiden Zeilen entfernen). Auf diese Weise müssen Sie die Veränderungen nicht bei jedem neuen Projekt wiederholt durchführen.

1 Rufen Sie die (MS-DOS-)Eingabeaufforderung auf. Sie finden sie im START-Menü von Windows unter PROGRAMME oder PROGRAMME/ZUBEHÖR.

2 Wechseln Sie mit dem Befehl `cd` (change directory) in das Verzeichnis, in dem sich die exe-Datei befindet. Mit dem Befehl `dir` können Sie den Inhalt des Verzeichnisses ausgeben.

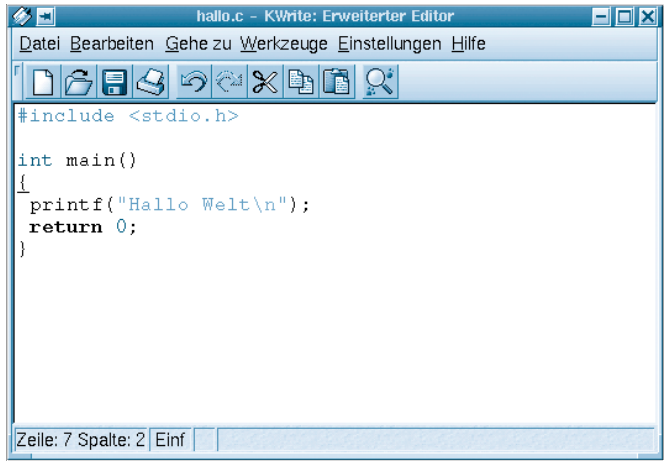
3 Rufen Sie das Programm aus der Konsole auf.



Verwendung des gcc-Compilers unter Linux

Als Beispiel zur Programmerstellung unter Linux wird hier der Kommandozeilen-Compiler *gcc* beschrieben. In der Regel ist dieser bereits auf Ihrem Linux-System installiert. Falls nicht, liegt er auf jeden Fall Ihrer Linux-Distribution bei.

1 Starten Sie einen Texteditor Ihrer Wahl (im Beispiel *KWrite*). Geben Sie das `Hallo Welt`-Programmbeispiel in den Editor ein und speichern Sie dieses in Ihrem Home-Verzeichnis unter dem Namen *hallo.c*.



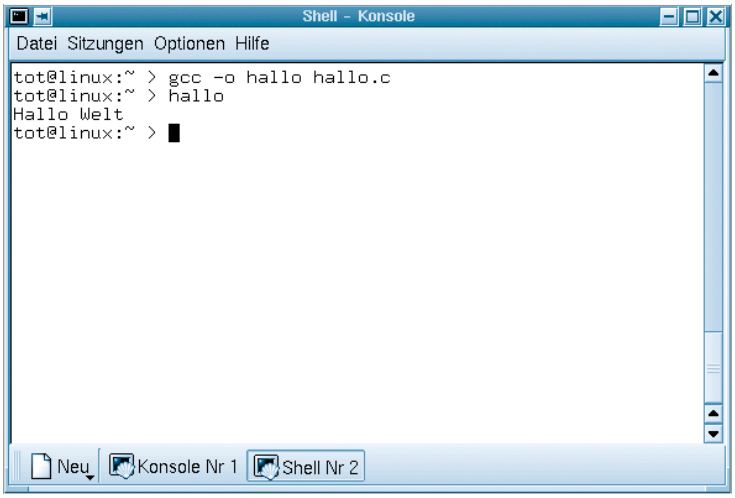
2 Öffnen Sie ein Konsolenfenster.

Wie das Konsolenfenster bei Ihnen aussieht und mit welchem Befehl Sie es aufrufen, hängt von der eingesetzten Linux-Distribution und dem verwendeten Window-Manager (KDE, Gnome etc.) ab. Meist befindet sich aber bereits nach der Installation ein Symbol zum Aufrufen der Konsole in der Taskleiste.

3 Wechseln Sie, falls nötig, in das Verzeichnis, in dem Sie die Datei *test.c* gespeichert haben.

Wie auch bei der Windows-Eingabeaufforderung können Sie in der Linux-Konsole das Verzeichnis mit dem Befehl `cd` wechseln. Beachten Sie dabei jedoch, dass die einzelnen Unterverzeichnisse unter Linux mit einem Slash `(/)` und nicht wie unter Windows mit einem Backslash `(\)` getrennt werden. Das Inhaltsverzeichnis können Sie mit dem Befehl `ls -l` auflisten.

4 Rufen Sie nun den gcc-Compiler von der Konsole mit dem Schalter `-o` auf und führen Sie das Programm aus.



Hinweis

Sollte sich das Programm nicht wie in diesem Beispiel mit dem Programmnamen starten lassen, versuchen Sie es, indem Sie dem Programmnamen die Zeichenfolge ./ voranstellen (was nichts anderes bedeutet, als einen Bezug zum aktuellen Verzeichnis herzustellen):

```
./hallo
```

Anmerkung zu anderen Compilern

Neben den hier vorgestellten Compilern gibt es natürlich noch eine Menge weiterer. Alle bekannten Programme auch nur kurz zu erwähnen, würde bereits zu weit führen und ist auch nicht Intention dieses Buches. Das Hauptziel dieses Kapitels liegt mehr darin, Ihnen einen Compiler für die gängigsten Betriebssysteme in den Grundzügen vorzustellen, damit Sie die Programmbeispiele eingeben und übersetzen können.

Natürlich lassen sich die Beispielquellcodes auch in Verbindung mit jedem anderen C- bzw. C++-Compiler verwenden. Die Übersetzung von Programmen läuft auf anderen – auch kommerziellen – Compilern in aller Regel sehr ähnlich ab.