# C++

## Programmieren mit einfachen Beispielen

DIRK LOUIS

Markt+Technik

→leicht →klar →sofort

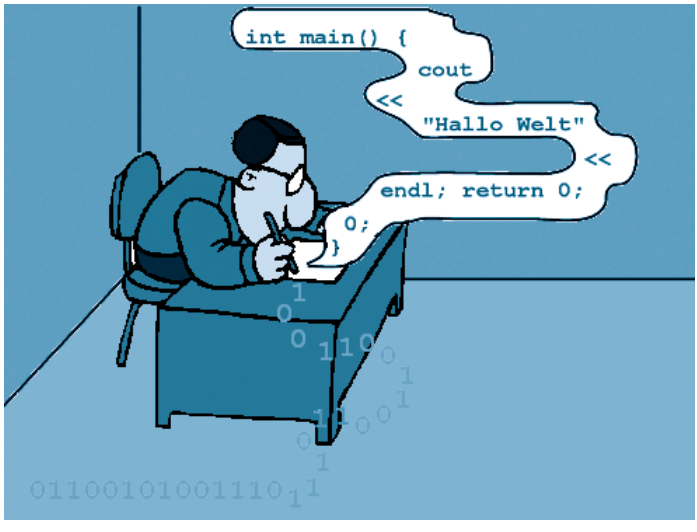
CD

Gleich loslegen mit dem Compiler

Auf Vista getestet

# Kapitel 3

# Wie erstellt man eigene Programme?




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

## Das können Sie schon:



## Das lernen Sie neu:

## Welcher Compiler darf es sein?

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

- den **Borland 5.5-Compiler** (auch C++Builder 5.5 genannt) für die Familie der **Windows**-Betriebssysteme

  Den Borland-Compiler können Sie kostenlos aus dem Internet herunterladen oder von der Buch-CD installieren (siehe Abschnitt »Programmerstellung mit dem Borland 5.5-Compiler (Windows)«).

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

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

Vielleicht möchten Sie aber auch mit einem ganz anderen Compiler arbeiten, beispielsweise einem Compiler mit integrierter Entwicklungsumgebung[2] wie Turbo C++, C++-Builder oder Microsoft Visual C++? Kein Problem! Sie können dieses Buch zusammen mit jedem Compiler lesen, der sich an den C++-ANSI-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 – für den Borland 5.5-Compiler wäre dies z.B. die Datei readme.txt im Installationsverzeichnis des Borland 5.5-Compilers. Sollte dies nicht fruchten, können Sie sich selbstverständlich auch an mich wenden. Ferndiagnosen zu fehlgeschlagenen Installationsversuchen sind zwar meist schwierig, aber so weit ich kann, helfe ich gerne.*

2 *Mehr zum Thema »integrierte Entwicklungsumgebungen« im letzten Abschnitt dieses Kapitels.*

# Programmerstellung mit dem Borland 5.5-Compiler (Windows)

Als Beispiel für einen leistungsfähigen und zuverlässigen Compiler für die Windows-Plattform möchte ich Ihnen die Kommandozeilenversion des Borland 5.5-Compilers vorstellen, den Sie auch auf Ihrer Buch-CD finden.

> **Hinweis**
>
> *Sollten Sie die Buch-CD verloren haben, können Sie den Compiler auch von der Website* `http://www.codegear.com` *herunterladen. (Zum Zeitpunkt der Drucklegung dieses Buchs mussten Sie von der Startseite aus auf den Link C++-Builder/free trial und dann auf Compiler (für Windows, Version 5.5) klicken.)*

## Installation

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

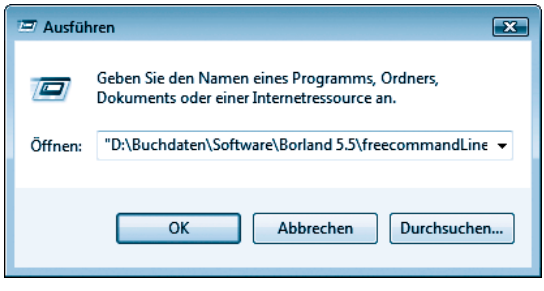



*Abbildung 3.1: Start des Installationsprogramms über Start/Alle Programme/Zubehör/Ausführen (bzw. Start/Ausführen)*

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

Öffnen Sie beispielsweise das Start-Menü und wählen Sie den Befehl *Ausführen* aus (unter Windows Vista ist der Befehl unter *Alle Programme/Zubehör* versteckt). Im gleichnamigen Dialogfenster tippen Sie den Pfad zum Installationsprogramm ein oder wählen die *freecommandLinetools.exe*-Datei über den Schalter *Durchsuchen* und den zugehörigen Suchdialog aus. Schicken Sie den Dialog mit einem Klick auf den *OK*-Schalter ab. Installie-

ren Sie den Compiler am besten in das vorgeschlagene Verzeichnis *C:\Borland\BCC55*.

> **Hinweis**
>
> *Unter Windows Vista kann es passieren, dass Ihnen am Ende der Installation eine Meldung angezeigt wird, die darauf hindeutet, dass die Installation fehlgeschlagen sei. Lassen Sie sich in diesem Fall nicht beunruhigen und bestätigen Sie per Klick, dass die Installation korrekt ist.*

## Konfiguration

Nach erfolgreicher Installation müssen Sie dem Compiler noch mitteilen, wo er seine Bibliotheken findet.

**3** Legen Sie dazu im *Bin*-Verzeichnis des Compilers zwei Textdateien *bcc32.cfg* und *ilink32.cfg* an.

Die Datei *bcc32.cfg* hat folgenden Inhalt:

```
-I"c:\Borland\Bcc55\include"
-L"c:\Borland\Bcc55\lib"
```

Die Datei *ilink32.cfg* enthält nur eine Zeile:

```
-L"c:\Borland\Bcc55\lib"
```

Ich habe beide Dateien bereits für Sie vorbereitet und im Verzeichnis *Software/Borland5.5* der Buch-CD gespeichert. So brauchen Sie die Dateien nur noch in Ihr *Bin*-Verzeichnis zu kopieren.

> **Hinweis**
>
> *Falls Sie den Compiler unter einem anderen Verzeichnis als c:\Borland\Bcc55 installiert haben, müssen Sie die Pfadangabe entsprechend anpassen!*



## Anpassung des Systempfads

Jetzt sollten Sie Ihr System noch so anpassen, dass Sie den Compiler möglichst bequem aufrufen können. Dazu tragen Sie das BIN-Verzeichnis des Compilers in Ihren Systempfad ein.

### Der Systempfad

*Der Systempfad ist eine Liste von Verzeichnissen, die das Betriebssystem bei Bedarf automatisch durchsucht.*

*Wenn Sie das BIN-Verzeichnis des Compilers nicht in den Systempfad einfügen, müssen Sie jedes Mal, wenn Sie den Compiler aus dem Konsolenfenster (Eingabeaufforderung) heraus aufrufen, vor dem Namen des Compilers (in unserem Fall* `bcc32`*) den kompletten Pfad zum Compiler angeben, also beispielsweise:*

```
C:\Borland\Bcc55\bin\bcc32
```

*Wenn das Verzeichnis mit der bcc32.exe-Datei des Compilers im Systempfad steht, genügt hingegen der Aufruf:* `bcc32`.

**4** Rufen Sie den Windows-Konfigurationsdialog für die Umgebungsvariablen auf und erweitern Sie den Wert der Systemvariablen PATH um den Eintrag `C:\Borland\Bcc55\bin;`.

Wie Sie den Konfigurationsdialog aufrufen, hängt von dem jeweiligen Windows-Betriebssystem ab.

Unter Windows NT/2000/XP/Vista rufen Sie über *Start* oder *Start/Einstellungen* die *Systemsteuerung* auf, schalten diese gegebenenfalls in die klassische Ansicht und gelangen via *System*, Register *Erweitert*[3], Schaltfläche *Umgebungsvariablen* zum Ziel. Danach können Sie die Systemvariable auswählen, zum Bearbeiten laden und den Pfad zum Borland-Compiler anhängen (siehe Abbildung 3.2).

3 *Unter Windows Vista der Link Erweiterte Systemeinstellungen*

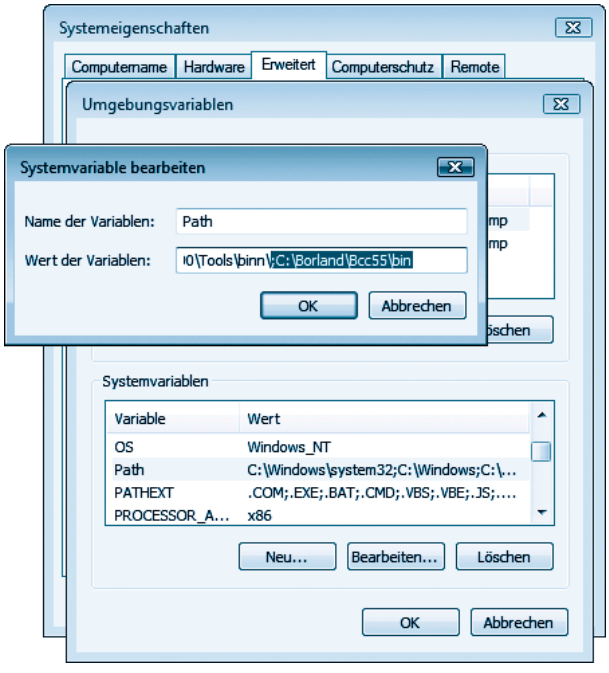



*Abbildung 3.2: Anpassung der PATH-Variablen unter Windows Vista (der neu hinzugekommene Eintrag ist markiert)*

Unter Windows ME lautet der Weg zum Konfigurationsdialog *Start/Programme/Zubehör/Systemtools/Systeminformationen*, Menü *Tools/System Konfiguration*, Register *Umgebung*.

Unter Windows 95/98 müssen Sie den Systempfad über einen passenden Eintrag in der Textdatei *C:\Autoexec.bat*-Datei erweitern. Legen Sie zur Sicherheit eine Kopie der Datei *C:\autoexec.bat* an und laden Sie die Datei danach in einen Editor (beispielsweise Notepad, Aufruf über *Start/Programme/Zubehör/Editor* oder über *Start/Ausführen, notepad* eingeben und abschicken). Fügen Sie am Ende der Datei folgende Zeile hinzu:

```
SET PATH=%PATH%;c:\Borland\Bcc55\bin;
```

**Hinweis**

*Falls Sie den Compiler unter einem anderen Verzeichnis als c:\Borland\Bcc55 installiert haben, müssen Sie die Pfadangabe entsprechend anpassen!*

### Testen

**5** Starten Sie den Rechner danach neu. (Ist unter Windows XP/Vista nicht nötig.)

**6** Öffnen Sie ein Konsolenfenster (*Start/Programme/MSDOS-Eingabeaufforderung* oder *Start/Programme/Zubehör/Eingabeaufforderung*).

**7** Tippen Sie `bcc32` ein und schicken Sie es durch Drücken der ↵-Taste ab.

In der Konsole sollte daraufhin eine Meldung des Compilers erscheinen.

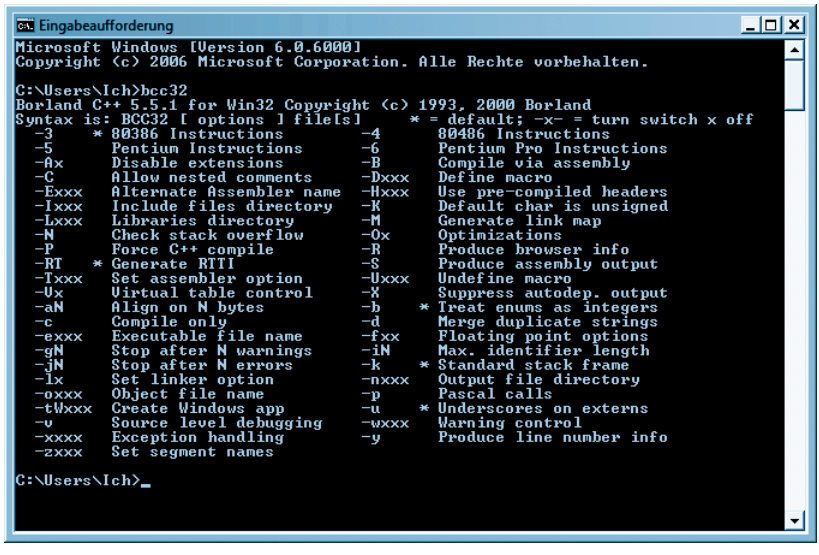



*Abbildung 3.3: Der Compiler ist korrekt installiert.*

Wenn Sie stattdessen eine Meldung der Form »Befehl oder Dateiname nicht gefunden« erhalten, haben Sie entweder den Compiler-Namen falsch eingetippt oder der PATH-Eintrag stimmt nicht.

## Programme erstellen und kompilieren

Um sich mit dem Compiler und den typischen Arbeitsabläufen beim Programmieren vertraut zu machen, werden Sie nun ein erstes kleines Programm erstellen. Der Quelltext des Programms sieht wie folgt aus:

```
// Hallo Welt-Programm

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

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

   return 0;
}
```

## Quelltext eingeben

**1** Starten Sie einen Texteditor Ihrer Wahl.

Sie suchen noch einen geeigneten Editor? Wie wäre es mit dem Windows-*Editor* (auch als *Notepad* bekannt), den Sie über das Start-Menü mit START/PROGRAMME/ZUBEHÖR/EDITOR aufrufen können.[4]

> **Hinweis**
>
> *Auch wenn konventionelle Texteditoren wie Editor oder Wordpad grundsätzlich vollkommen ausreichen, lassen sie doch verschiedene Extras missen, die gerade dem Programmierer die Arbeit sehr erleichtern können – wie z.B. die automatische Syntaxhervorhebung oder die Anzeige der Zeilennummern[5]. Wer auf diese Funktionen nicht verzichten möchte, muss zu einem Editor greifen, der sich auf die Quelltextverarbeitung spezialisiert hat. Empfehlenswerte Vertreter dieser Gattung sind z.B. der emacs für Linux (gehört in der Regel zum Lieferumfang) und der Notepad++-Editor für Windows (kann von* `http://notepad-plus.sourceforge.net/de/site.htm` *heruntergeladen werden).*

**2** Tippen Sie den Quelltext genauso ab, wie er oben aufgelistet ist. Was der Code im Einzelnen bedeutet, werden Sie im nächsten Kapitel erfahren.

---

4 *Alternativ können Sie Start/Ausführen aufrufen, in dem erscheinenden Dialogfenster* `notepad.exe` *eingeben und abschicken. Unter Windows Vista tippen Sie* `notepad.exe` *direkt in das Suchfeld des Start-Menüs ein.*

5 *Der Compiler gibt Meldungen zu Syntaxfehlern immer unter Angabe der betroffenen Zeilennummer aus. Wenn Sie einen Editor mit Zeilennummerierung verwenden, müssen Sie nicht selbst zählen, um die verdächtige Zeile zu finden.*

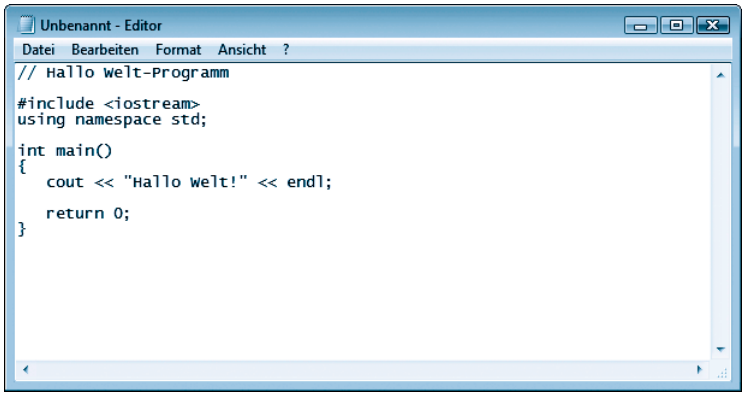



*Abbildung 3.4: Zum Eintippen der Quelltexte genügt ein einfacher Editor*

**Achtung**

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

**3** Speichern Sie das Programm mit der Extension .cpp.

Um die weiteren Schritte leichter nachvollziehen zu können, sollten Sie auf Ihrer Festplatte unter dem Laufwerk *C:* ein Verzeichnis *MeineProjekte* und darunter ein Unterverzeichnis *Test* anlegen. In diesem Unterverzeichnis speichern Sie dann die Quelltextdatei als *Test.cpp*.

**Hinweis**

*Bei Verwendung von Notepad gibt es manchmal Probleme, weil der Notepad-Editor die Dateiendung .txt an die gespeicherten Dateien anhängt (aus Dateiname.cpp wird dann Dateiname.cpp.txt). Um dies zu vermeiden, gibt es zwei Möglichkeiten. Die erste Lösung besteht darin, den kompletten Dateinamen, samt Extension, in Anführungszeichen zu setzen: "Dateiname.cpp". Die zweite Möglichkeit ist, die Extension .cpp im Windows Explorer zu registrieren. Speichern Sie dazu nach Methode 1 eine Datei mit der Extension .cpp. Wechseln Sie danach in den Windows Explorer und doppelklicken Sie auf die Datei. Ist die Extension noch nicht registriert, erscheint jetzt ein Dialogfenster (oder eine Folge von Dialogfenstern), in denen Sie Notepad als*

*gewünschtes Bearbeitungsprogramm auswählen und die Option setzen können, die dafür sorgt, dass dieser Dateityp immer mit dem ausgewählten Programm geöffnet wird. Wenn Sie anschließend den Dialog mit OK abschicken, wird die Dateiendung .cpp registriert und mit Notepad als Standardverarbeitungsprogramm verknüpft. Danach können Sie CPP-Dateien per Doppelklick in Notepad laden und werden nie wieder Ärger mit an CPP-Dateien angehängten .txt-Dateiendungen haben.*

## Kompilieren

Bis jetzt haben wir im Grunde 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 wenige Minuten von unserem ersten eigenen Programm entfernt sind.

**4** Öffnen Sie zum Kompilieren ein Konsolenfenster.

Viele Windows-Anwender sind heutzutage gar nicht mehr mit dem Umgang mit der Konsole vertraut. Unter Windows heißt die Konsole je nach Version »Konsole« oder »Eingabeaufforderung« und wird über das Start-Menü aufgerufen (*Start/Programme/Eingabeaufforderung* bzw. *Start/Programme/Zubehör/Eingabeaufforderung*).

Die Konsole, siehe auch Abbildung 3.5, ist ein zeilenorientiertes Befehlseingabefenster, in deren jeweils letzten Zeile Sie einen Systembefehl eingeben und mit der ↵-Taste abschicken können. Beachten Sie auch den »Prompt« zu Beginn der Zeile.

Der Prompt zeigt Ihnen an, dass die Konsole auf die Eingabe eines Befehls wartet. Das Aussehen des Prompts ist veränderbar. Meist ist er so konfiguriert, dass der Pfad zu dem aktuellen Verzeichnis und ein abschließendes > angezeigt werden. Hinter dem > tippen Sie Ihren Befehl ein.

```
C:\Pfad> IhrBefehl
```

**Hinweis**

*Eine kurze Einführung in die Bedienung der Konsole finden Sie als Tutorial auf der Website* `www.carpelibrum.de`.

**5** Wechseln Sie in der Konsole mithilfe des `cd`-Befehls in das Verzeichnis, in dem das Programm steht (beispielsweise `cd c:\MeineProjekte\Test`).

| cd-Befehl | Beschreibung |
|---|---|
| E: | Wechsel zum Laufwerk *E:* |
| cd .. | Wechsel in das übergeordnete Verzeichnis |
| cd Verzeichnis | Wechsel in das angegebene untergeordnete Verzeichnis |
| cd C:\Projekte\Verzeichnis | Wechsel in das angegebene Verzeichnis |

*Tabelle 3.1: cd-Befehle der Konsole*

**6** Kompilieren Sie die Datei mit folgendem Befehl:

```
bcc32 Test.cpp
```

Der Compiler beginnt nun mit der Übersetzung des Quelltexts in Maschinencode. Stößt er dabei auf syntaktische Fehler im Quelltext endet die Kompilierung mit einer Liste von Fehlermeldungen. Wechseln Sie dann zurück in den Editor, prüfen Sie die Textzeilen, auf die die Fehlermeldungen des Compilers verweisen, und korrigieren Sie gefundene Fehler. Speichern Sie Ihre Änderungen und kompilieren Sie erneut.

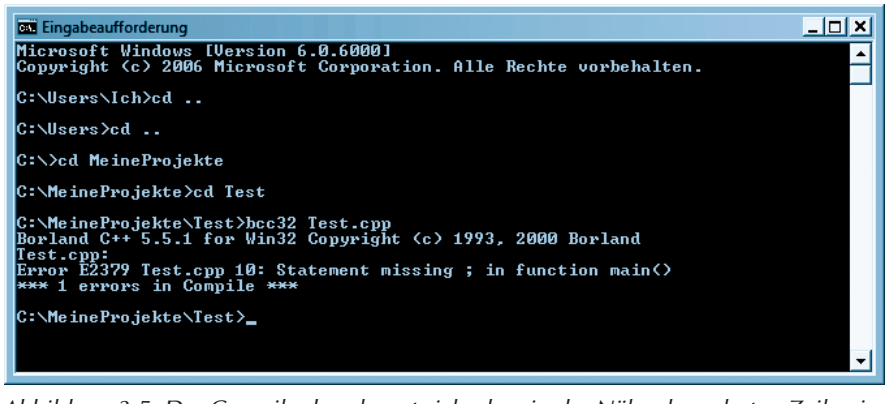



*Abbildung 3.5: Der Compiler beschwert sich, dass in der Nähe der zehnten Zeile ein Semikolon fehlt.*

**Hinweis**

*Grämen Sie sich nicht, wenn Sie die vom Compiler bemängelten Fehler nicht aufspüren können. Wenn einem alles fremd scheint, ist es manchmal sehr schwer, richtig und falsch zu unterscheiden. Im gleichen Maße aber, in dem Sie mehr und mehr über C++ und seine Syntaxformen lernen, wird sich auch Ihre Fähigkeit, Fehler aufzuspüren, verbessern. Daher ist es auch nicht weiter schlimm, wenn Sie jetzt ein bisschen mogeln und den Quelltext einfach aus der Datei Test.cpp von der Buch-CD kopieren.*

*Im Übrigen kommt es eher selten 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 sogenannten »syntaktischen Fehler« noch recht harmlos, denn sie werden – im Gegensatz zu den logischen Fehlern – vom Compiler entdeckt und meist recht gut lokalisiert.*

Nur wenn der Quelltext fehlerfrei ist, übersetzt ihn der Compiler in Maschinencode und speichert diesen in einer eigenen Datei mit der Extension *.obj*. Danach ruft er den Linker auf, der aus dem Maschinencode in der OBJ-Datei eine ausführbare EXE-Datei macht.

**Hinweis**

*Wenn Sie nach erfolgreicher Kompilierung einen Blick in das Programmverzeichnis werfen (innerhalb der Konsole brauchen Sie dazu nur den Befehl* `dir` *abzuschicken und schon wird der Inhalt des aktuellen Verzeichnisses aufgelistet), sehen Sie dort neben der CPP-Quelltextdatei und der EXE-Datei des Programms auch die erzeugte Objektdatei und eine Datei mit der Endung .tds. Die OBJ- und die TDS-Datei sind »Abfallprodukte« der Kompilierung. Sie können sie löschen.*

### Ausführen und 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.

7 Führen Sie das Programm in der Konsole aus. Tippen Sie den Namen der EXE-Datei ein und schicken Sie mit ↵ ab.

Wenn alles gut geht, erscheint auf der Konsole die Ausgabe: »Hallo Welt!«


*Abbildung 3.6: Erfolgreiche Ausführung in einem Konsolenfenster*

## 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 Hinweis, 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-CD nachinstalliert werden.

## Programme erstellen und kompilieren

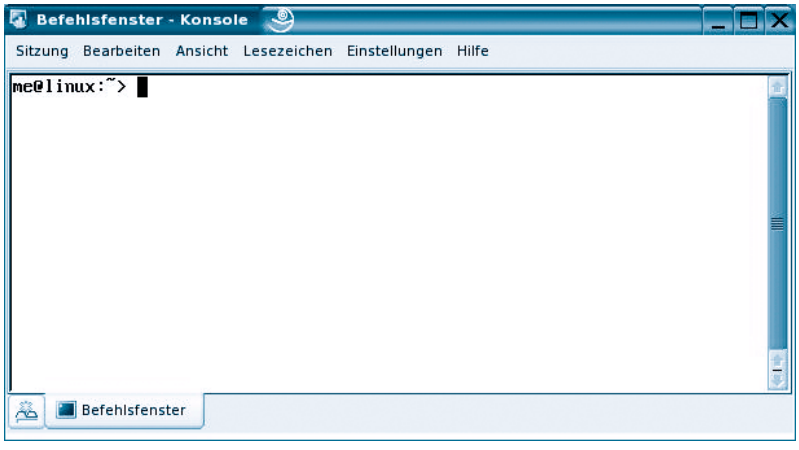



*Abbildung 3.7: Konsolenfenster unter Linux*

**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. Unter KDE können Sie Konsolenfenster beispielsweise über die KDE-Taskleiste aufrufen (Symbol *Terminal-Programm*).

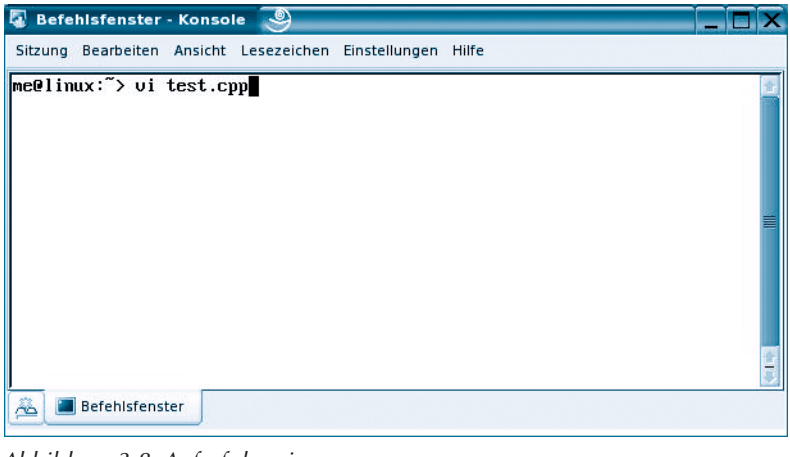



*Abbildung 3.8: Aufruf des vi*

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

Statt des *vi* können Sie auch jeden beliebigen anderen ASCII-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.

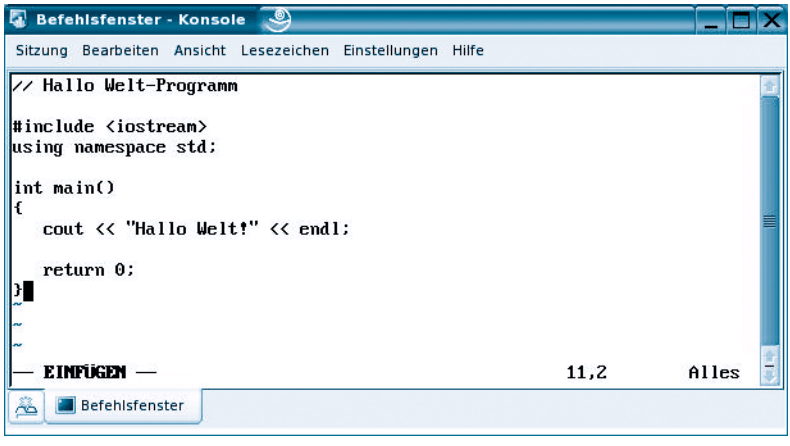



*Abbildung 3.9: Quelltext eintippen*

**4** Geben Sie den Programmquelltext ein. (Wenn Sie den Quelltext in der Abbildung nicht lesen können, blättern Sie zurück zum Abschnitt »Programme erstellen und kompilieren« für den Borland-Compiler, 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.

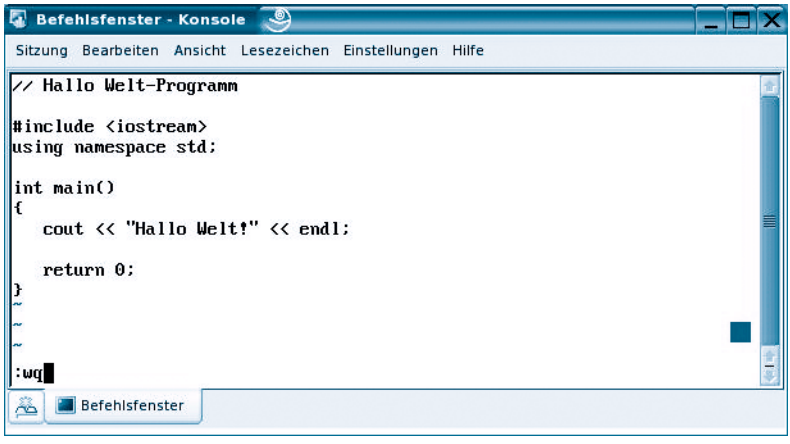



*Abbildung 3.10: Speichern und beenden*

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

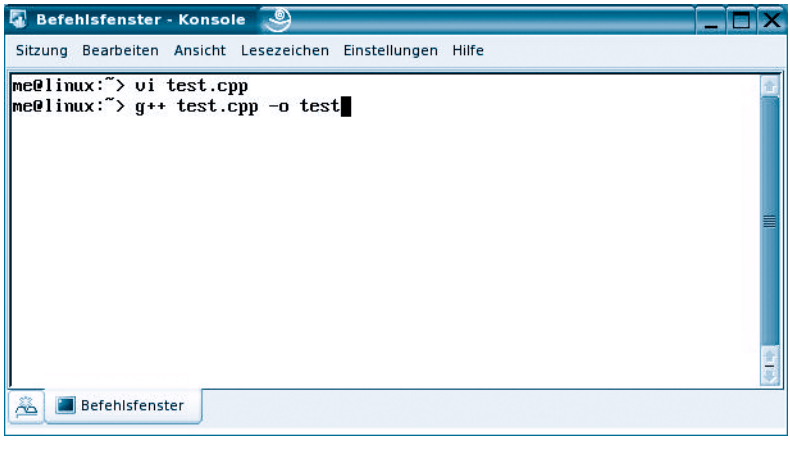



*Abbildung 3.11: Kompilieren mit dem g++-Compiler*

**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.

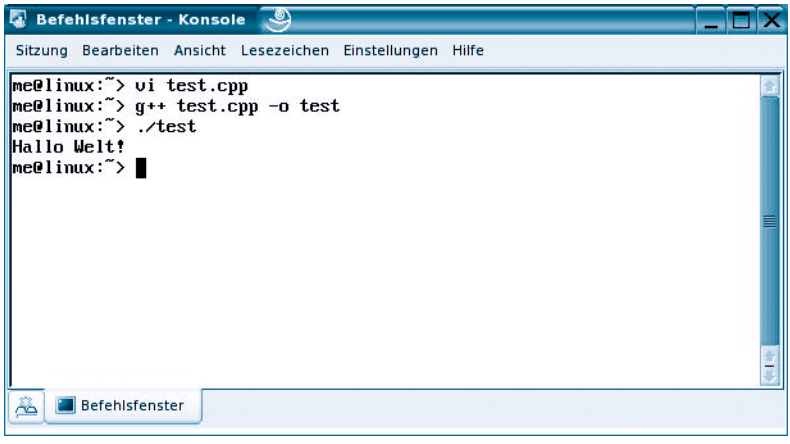



*Abbildung 3.12: Ausführung in einem Konsolenfenster*

**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 Testprogramms 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.

# Einrichtung einer eigenen Entwicklungsumgebung

Als Erstes sollten Sie sich überlegen, wie Sie die Dateien Ihrer Programme auf der Festplatte verwalten wollen.

## Organisation der Quelldateien

Eine Möglichkeit wäre, unter einem eigenen übergeordneten Verzeichnis für die einzelnen Programme Unterverzeichnisse anzulegen. Zum Nachvollziehen der Beispiele in diesem Buch bietet es sich allerdings an, unter dem übergeordneten Verzeichnis Unterverzeichnisse für die Buchkapitel anzulegen.

Wenn Sie das übergeordnete Verzeichnis *MeineProjekte* nennen, könnte Ihre Verzeichnisstruktur wie folgt aussehen:

*C:\MeineProjekte*

*C:\MeineProjekte\Kap02*

*C:\MeineProjekte\Kap03*

*C:\MeineProjekte\Kap04*

*...*

Unter den Verzeichnissen für die einzelnen Kapitel legen Sie dann Verzeichnisse für die Programme an und in diesen speichern Sie die Quelldateien der Programme.

**1** Richten Sie jetzt auf Ihrem Rechner ein Verzeichnis *MeineProjekte* ein.

Unter Windows können Sie das Verzeichnis direkt unter *C:* anlegen, also als *C:\MeineProjekte*. (Unterverzeichnisse können im Windows Explorer, Aufruf mit [WinBef]+[E], erzeugt werden.)

Unter Linux legen Sie das Verzeichnis unter Ihrem Home-Verzeichnis an.

**2** Legen Sie ein Unterverzeichnis *Kap04* für Kapitel 4 an.

## Organisation der benötigten Werkzeuge

Jetzt sollten Sie Ihren Desktop noch so gestalten, dass Sie alle für die Programmierung typischen Aufgaben effizient erledigen können. Ausgangspunkt sei ein leerer Desktop, schließen Sie also gegebenenfalls alle aktuell geöffneten Fenster.

**1** Öffnen Sie ein Ordnerfenster (Windows Explorer o.a.).

**2** Legen Sie im Ordnerfenster unter dem Verzeichnis *C:\MeineProjekte\Kap04* ein Unterverzeichnis *Hallo* für das nächste zu erstellende Programm an.

Das Ordnerfenster benötigen Sie, um das Verzeichnis und die Quelltextdatei(en) des aktuellen Programms im Blick zu behalten. Außerdem können Sie im Ordnerfenster bei Bedarf weitere Kapitel- oder Programmverzeichnisse anzulegen und bestehende Quelldateien zur Ansicht oder Bearbeitung öffnen.

**3** Rufen Sie den Texteditor Ihrer Wahl auf.

**4** Legen Sie eine neue Quelltextdatei namens *Hallo.cpp* an (die Datei darf vorerst ruhig leer bleiben) und speichern Sie diese im Programmverzeichnis *C:\MeineProjekte\Kap04\Hallo*.

**5** Öffnen Sie ein Konsolenfenster.

**6** In der Konsole wechseln Sie mithilfe des *cd*-Befehls (»change directory«) in das Programmverzeichnis, in dem die zu kompilierende Quelldatei steht.

Ihr Desktop sollte nun ungefähr wie in Abbildung 3.13 aussehen.

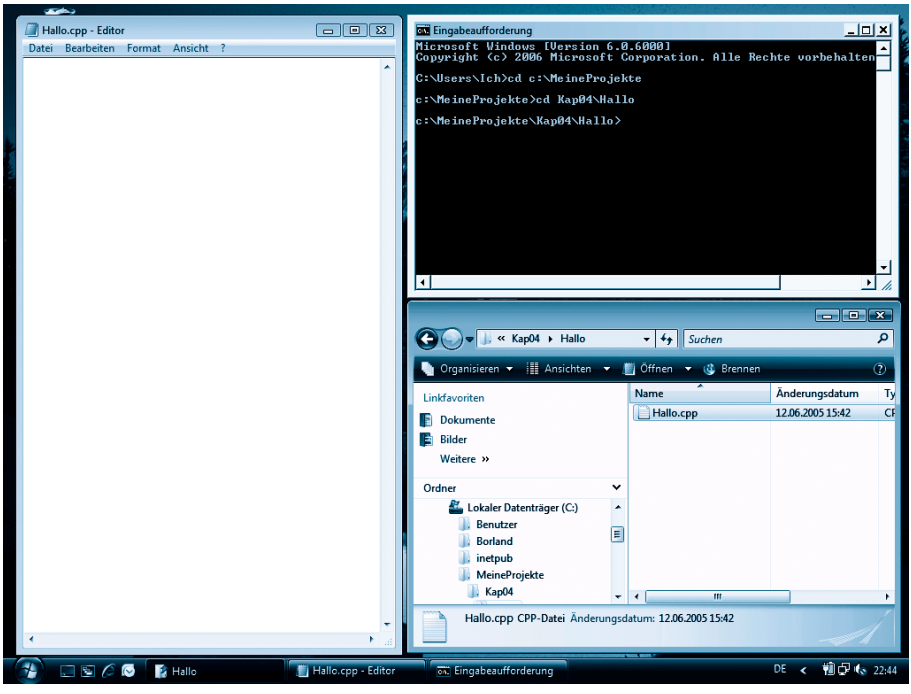



*Abbildung 3.13: Desktop mit Editor, Konsole und Explorer (unter Windows Vista)*

Im Editor setzen Sie Ihre Quelltexte auf. Den fertig bearbeiteten Quelltext speichern Sie im zugehörigen Programmverzeichnis.

Nach dem Speichern wechseln Sie in die Konsole, von wo aus Sie das Programm kompilieren und ausführen.

Treten bei der Kompilierung Fehler auf, korrigieren Sie die entsprechenden Zeilen im Quelltext, speichern, wechseln zur Konsole und kompilieren erneut. Die Konsolenbefehle müssen Sie dazu nicht einmal neu eingeben. Die Konsole merkt sich alle bereits abgeschickten Befehle in ihrer History, die Sie mithilfe der Pfeiltasten durchgehen können. (Achtung, Windows 98-Anwender! Für die Win98-Konsole muss die History explizit durch Abschicken des Befehls `doskey` aktiviert werden.)

Damit sind Sie gerüstet und das Abenteuer kann beginnen!

# Integrierte Entwicklungsumgebungen

Die Arbeit mit reinen Compilern und anderen Programmierwerkzeugen, die nur von der Konsole aus zu bedienen sind, ist etwas unhandlich. Viele Programmierer greifen daher zu integrierten Entwicklungsumgebungen (IDE), deren grafische Benutzeroberfläche den Komfort eines übergeordnetes Bedienpults bietet, von dem aus sämtliche Programmierarbeiten bequem erledigt werden können. Die meisten integrierten Entwicklungsumgebungen warten daher nicht nur mit Menübefehlen zum Kompilieren und Ausführen der Programme auf, sondern auch mit

- integrierten Editoren, die speziell für die Erstellung von Quelltexten ausgelegt sind und den Programmierer mit Optionen wie Zeilennummerierung, Syntaxhervorhebung, automatischer Zeileneinzug, Codevervollständigung und Einblendung von Hilfetexten erfreuen,
- integriertem Debugger zum Aufspüren von Laufzeitfehlern,
- einer ausgeklügelten Projektverwaltung, die dem Programmierer dabei hilft, die Dateien seiner Programme (Quelltextdateien, kompilierte *.obj*-Dateien, Ressourcen wie Bild- oder Sounddateien) übersichtlich zu verwalten,
- vorgefertigten Programmgerüsten und
- visueller Programmierung. (Betrifft vor allem die Erzeugung von Anwendungen mit grafischer Benutzeroberfläche (GUI). Die Fenster der Benutzeroberfläche können in einem grafischen Editor aufgebaut und bearbeitet werden, den zugehörigen C++-Code erzeugt die Entwicklungsumgebung automatisch.)

Die wichtigsten C++-Entwicklungsumgebungen sind wohl Microsoft Visual C++, C++-Builder, Eclipse und KDevelop.

Eclipse ist kostenfrei. Sie müssen aber zusätzlich zu der reinen Eclipse-IDE auch noch ein passendes Modul für die C++-Programmierung installieren. Herunterladen können Sie Eclipse von der Website `www.eclipse.org`.

Vom C++-Builder gibt es Trial-Versionen, die Sie über die Website `www.codegear.com` beziehen können.

Der Microsoft Visual C++-Compiler kann derzeit sogar in der Version »Visual C++ 2005 Express Edition« kostenlos von der Website `www.microsoft.com/germany/msdn/vstudio/products/express/visualc/default.mspx` heruntergeladen werden.

KDevelop ist eine integrierte Entwicklungsumgebung für Linux/KDE.

So verführerisch der Gebrauch einer IDE aber auch sein mag, möchte ich Ihnen dennoch raten, anfangs auf deren Einsatz zu verzichten und mit einem der oben besprochenen reinen Compiler zu arbeiten. Gerade weil diesen der Komfort einer integrierten Entwicklungsumgebung und die Leichtigkeit und trügerische Sicherheit der visuellen Programmierung fehlt, halte ich sie für den Einstieg ideal. Ich möchte, dass Sie erst einmal selbst lernen, wie man gute und sichere Programme schreibt und sich nicht gleich von Anfang an auf eine Entwicklungsumgebung verlassen, die Ihnen zwar vieles abnimmt, aber auch vieles vor Ihnen verbirgt. Sie sollen ja nicht aufs Geratewohl programmieren, sondern verstehen, was Sie programmieren. Später können Sie dann jederzeit auf eine integrierte Entwicklungsumgebung umsteigen.

# Rätselhaftes C++

Wer sind wir? Woher kommen wir? Wohin gehen wir? Nichts steht isoliert in der Welt. Auch nicht die Programmiersprache C++. Einen der Ursprünge von C++ haben Sie bereits kennengelernt: 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.