PEARSON Education

Markt+Technik

easy

C++

Programmieren mit einfachen Beispielen

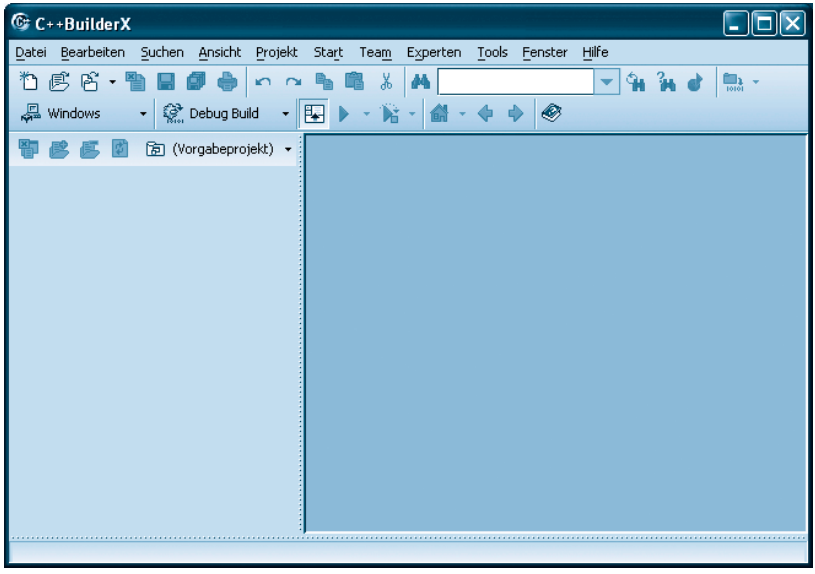
DIRK LOUIS

Markt+Technik

→ leicht → klar → sofort

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.

Das können Sie schon:

Das lernen Sie neu:

Welcher Compiler darf es sein?

Stellvertretend für viele andere leistungsfähige Compiler mit integrierter Entwicklungsumgebung stelle ich Ihnen hier den C++BuilderX-Compiler von Borland vor, den Sie von der Buch-CD aus installieren können. Der C++BuilderX kann unter Windows XP, Windows 2000 sowie RPM-basierten Linux-Systemen (vorzugsweise RedHat) installiert werden.

Sollten Sie keines dieser Betriebssysteme Ihr eigen nennen oder feststellen, dass die Komplexität des C++BuilderX Sie eher abschreckt, als Ihnen Lust auf das Programmieren zu machen, müssen Sie deshalb nicht verzweifeln:

- Wenn Sie mit einem Windows-Betriebssystem arbeiten, beispielsweise Windows 98 oder Windows Me, können Sie auf den Borland 5.5-Compiler (auch C++Builder 5.5 genannt) zurückgreifen, den Sie kostenlos aus dem Internet herunterladen oder von der Buch-CD installieren Buch-CD
- können. (Siehe Abschnitt »Programmerstellung mit dem reinen Borland 5.5-Compiler (Windows)«.)
- Wenn Sie Linux-Anwender sind, wird auf Ihrem System höchstwahrscheinlich bereits der GNU-C++-Compiler g++ 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 dem Watcom-Compiler oder dem Microsoft Visual C++-Compiler, weil einer Ihrer Freunde oder Kollegen bereits mit diesem Compiler arbeitet. 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.

Für welchen Compiler auch immer Sie sich entscheiden, überfliegen Sie trotzdem die Ausführungen zum C++BuilderX, da diese neben compilerspezifischen Schritt-für-Schritt-Anleitungen auch Hinweise von allgemeiner Gültigkeit enthalten (Beachtung der Groß-/Kleinschreibung, Umgang mit Compiler-Meldungen etc.).

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 – install.html *im* C++BuilderX-Verzeichnis *auf der Buch-CD bzw.* readme.txt *im Installationsverzeichnis des Borland 5.5-Compilers. Sollte dies nicht fruchten, können Sie sich selbstverständlich auch an mich wenden (dirk@carpelibrum.de). Ferndiagnosen zu fehlgeschlagenen Installationsversuchen sind zwar meist schwierig, aber soweit ich kann, helfe ich gerne.*

Programmerstellung mit dem C++BuilderX-Compiler (Win2000/XP/Linux)

Als Beispiel für einen Compiler mit integrierter Entwicklungsumgebung stelle ich Ihnen hier den C++BuilderX vor, den Sie von der Buch-CD installieren können.

Installation

Bevor Sie mit der Installation des C++BuilderX, Personal Edition, beginnen, sollten Sie sicher sein, dass diese Version für Sie geeignet ist. Um den C++BuilderX von der Buch-CD installieren und verwenden zu können, müssen Sie

- mit Windows 2000 oder Windows XP arbeiten
- über ungefähr 300-500 MByte freien Festplattenspeicher verfügen (ca. 700 MByte für Linux-Anwender oder Sie müssen sich zuerst eine Installations-CD brennen)
- Internetanschluss haben, um den Aktivierungsschlüssel anzufordern (der an eine von Ihnen anzugebende E-Mailadresse gesendet wird)
- Linux-Anwender müssen sicherstellen, dass der GNU-C++-Compiler installiert ist. (Von Konsole aus g++ eingeben. Ist der Befehl unbekannt, müssen Sie g++ installieren, siehe Abschnitt »Programmerstellung mit dem g++-GNU-Compiler (Linux)«.)

Wenn Sie diese Bedingungen erfüllen, können Sie mit der Installation beginnen:

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

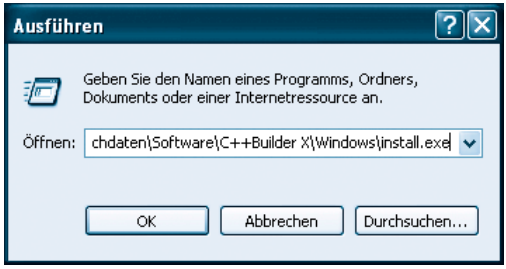


Abbildung 3.1: Start des Installationsprogramms über Start/Ausführen.

2 Starten Sie das Installationsprogramm.

Windows-Anwender öffnen das Start-Menü und wählen den Befehl *Ausführen* aus. Im gleichnamigen Dialogfenster tippen Sie den Pfad zum Installationsprogramm des C++BuilderX ein (*CD-ROM:\Buchdaten/Software/C++Builder X*) oder wählen die *install.exe*-Datei über den Schalter *Durchsuchen* und den zugehörigen Suchdialog aus. Schicken Sie den Dialog mit einem Klick auf den *OK*-Schalter ab.

Für Linux-Anwender ist es etwas schwieriger, weil die Download-Datei nur noch in gepackter Form auf der Buch-CD Platz fand. Sie finden die Original-Downloaddatei im *Buchdaten/Software/C++Builder X/Linux*-Verzeichnis der Buch-CD.

Entpacken Sie diese Datei in ein beliebiges Verzeichnis (nach erfolgreicher Installation können Sie dieses Verzeichnis wieder löschen). Öffnen Sie ein Konsolenfenster, wechseln Sie in das gerade entpackte Unterverzeichnis *Linux* und schicken Sie den Befehl `./install.bin` ab, um das Installationsprogramm zu starten.

> **Hinweis**
>
> *Die nachfolgenden Screenshots illustrieren die Installation und die Arbeit mit dem C++BuilderX. Die Screenshots wurden unter Windows XP geschossen, gelten aber auch für Linux und andere Windows-Betriebssysteme.*

Es springen nun nacheinander verschiedene Installationsfenster auf, die nach einer Weile wieder verschwinden. Schließlich erscheint das *Einführung*-Fenster, in dem Sie auf *Weiter* klicken.

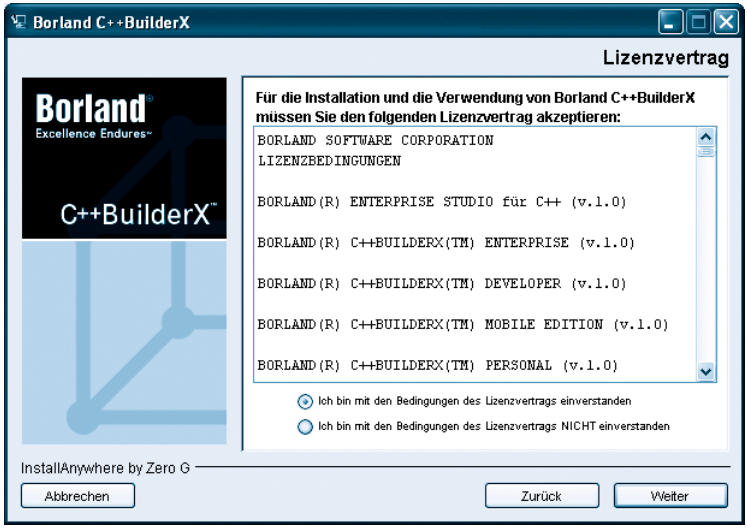


Abbildung 3.2: Der Lizenzvertrag

3 Lesen Sie den Lizenzvertrag. Nachdem Sie Ihr Einverständnis erklärt haben (obere Option anklicken), können Sie die Installation mit *Weiter* fortsetzen.

Sie kommen zur Auswahl der zu installierenden Produktkomponenten.

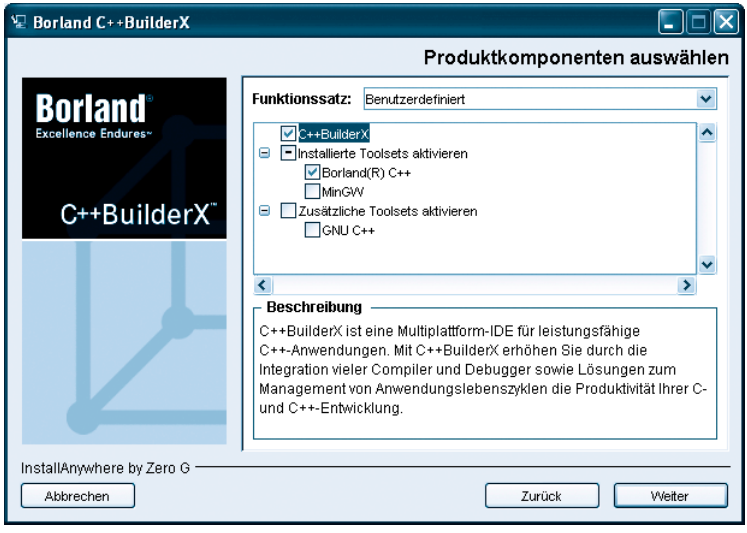


Abbildung 3.3: Welche Komponenten sollen installiert werden?

4 Im Dialog der Produktkomponenten deaktivieren Windows-Anwender die Optionen *MinGW* und *Zusätzliche Tools aktivieren*. Linux-Anwender deaktivieren die Option *Installierte Toolsets aktivieren*. Anschließend klicken Sie auf *Weiter*.

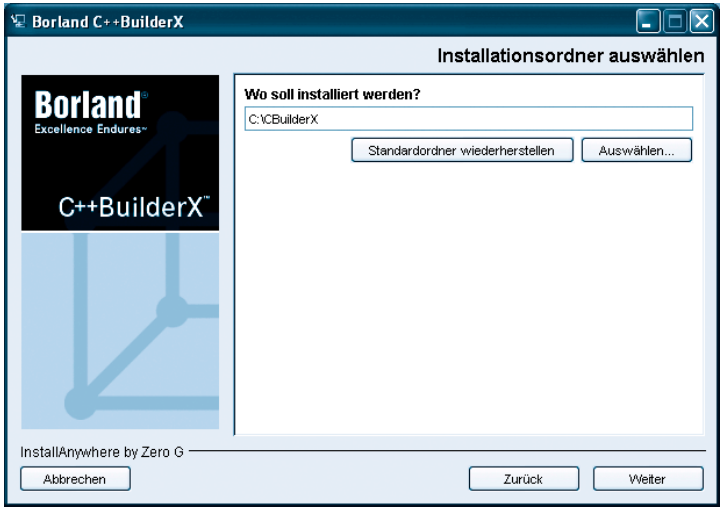


Abbildung 3.4: Festlegung des Installationsverzeichnisses

5 Im fünften Dialog können Sie festlegen, in welches Verzeichnis der C++BuilderX installiert werden soll.

Wenn nichts dagegen spricht, behalten Sie einfach die Vorgabe bei.

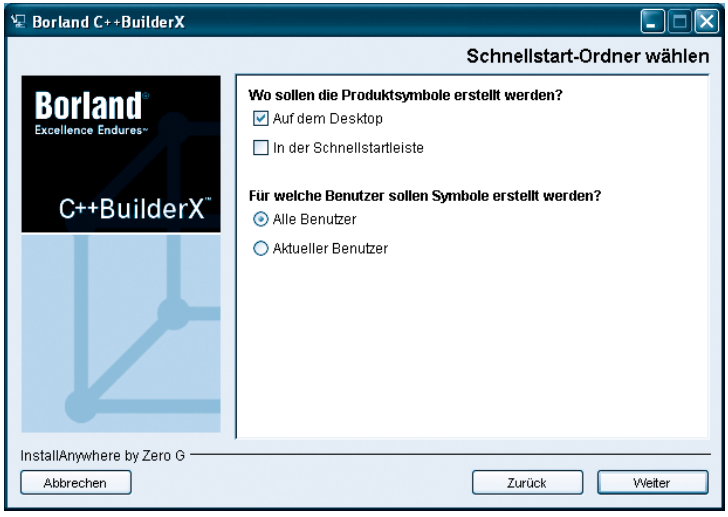


Abbildung 3.5: Aufruf mit einem Klick

6 Im sechsten Dialog können Sie festlegen, wo und für welche Benutzer die Produktsymbole angezeigt werden sollen.

Wenn nichts dagegen spricht, behalten Sie auch hier einfach die Vorgabe bei.

7 Im letzten Dialog werden Ihre Installationsdaten noch einmal zusammengefasst. Starten Sie nun die eigentliche Installation, indem Sie auf *Installieren* drücken.

Wenn alle Dateien kopiert sind, ist Ihre Festplatte um ca. 150 MByte ärmer und Sie um eine C++BuilderX-Version reicher. Ein allerletzter Dialog zeigt Ihnen an, dass die Installation erfolgreich abgeschlossen wurde.

Nachdem die Installation abgeschlossen ist, müssen Sie die Software noch aktivieren. Borland stellt Ihnen den C++BuilderX in der Personal Edition zwar kostenlos zur Verfügung, möchte aber dafür, dass Sie sich online registrieren und einen kurzen Fragebogen ausfüllen.

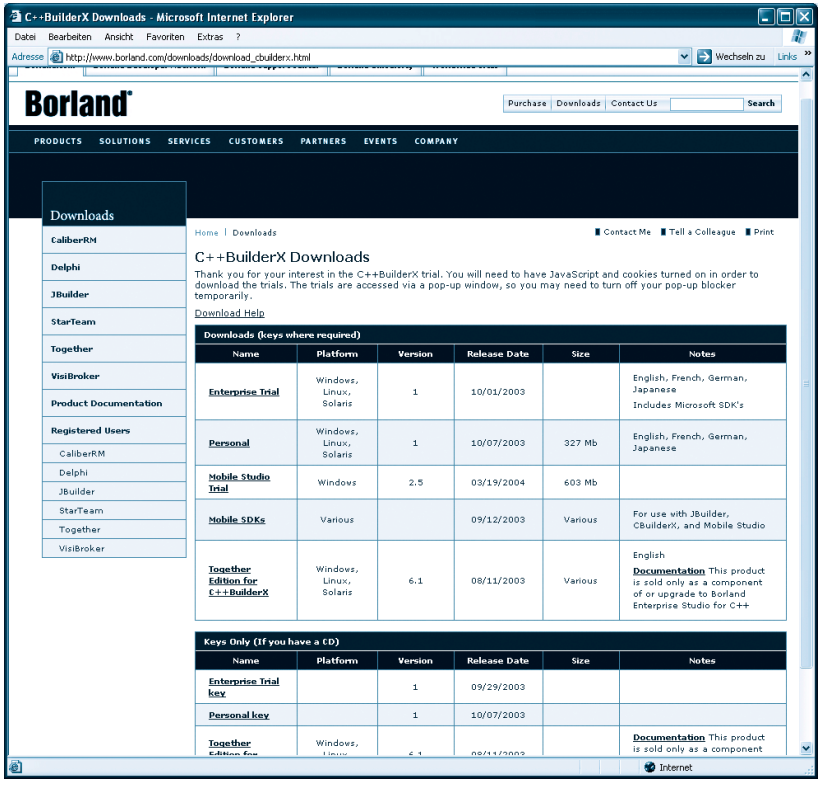


Abbildung 3.6: Auf der C++BuilderX-Download-Seite gibt es auch den Aktivierungsschlüssel.

8 Stellen Sie eine Verbindung zum Internet her und steuern Sie die Webseite `http://www.borland.com/downloads/download_cbuilderx.html` an. In der unteren Tabelle finden Sie einen Link »Personal Key«, den Sie anklicken.

> **Hinweis**
>
> *Falls Borland seine Website in der nächsten Zukunft umstrukturieren sollte, sodass die Webseite nicht mehr unter der angegebenen Adresse zu finden ist, gehen Sie einfach zu* `http://www.borland.de` *und folgen Sie den Links zum Download-Bereich und zum C++BuilderX.*

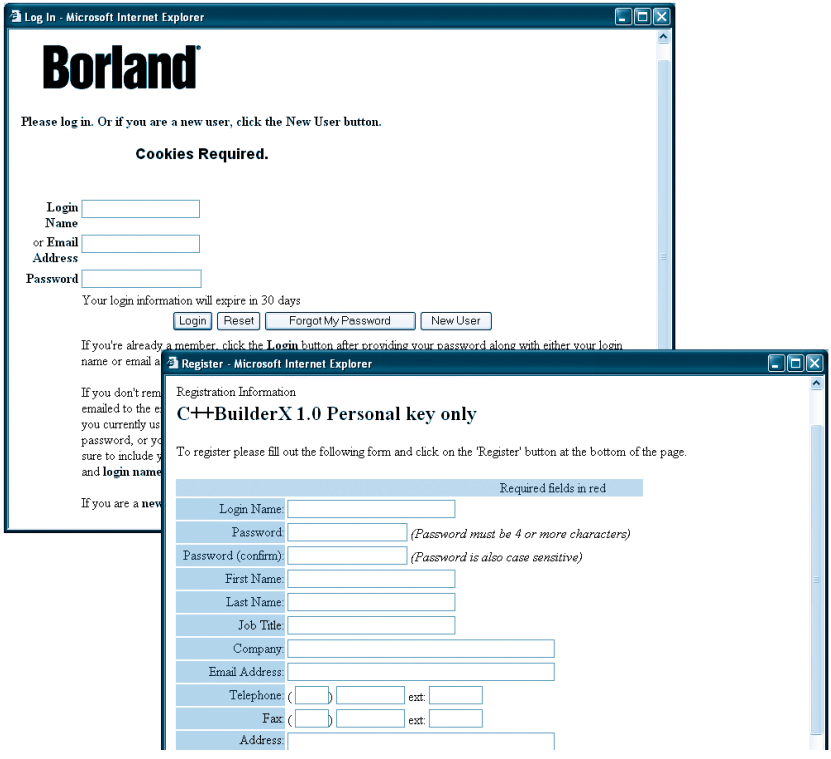


Abbildung 3.7: Ohne Registrierung kein Aktivierungsschlüssel

9 Wenn Sie bereits Mitglied der Borland-Community sind, loggen Sie sich mit Ihrem Benutzernamen und Passwort ein. Andernfalls drücken Sie auf den Schalter *New User* und füllen das Anmeldeformular aus.

Bis auf »Job Title«, »Fax« und »Address« müssen Sie alle Felder ausfüllen. Notieren Sie sich Ihren Benutzernamen (»Login Name«) und Ihr Passwort für

den Fall, dass Sie irgendwann wieder einmal etwas von der Borland-Site herunterladen möchten. Mogeln Sie nicht bei der E-Mail-Adresse! An diese Adresse wird der Aktivierungsschlüssel für den C++BuilderX gesendet. Wählen Sie am unteren Ende als Sprache (»Language«) *German* aus und drücken Sie dann den *Register*-Schalter.

10 Anschließend werden Sie möglicherweise noch aufgefordert, ein Frageformular auszufüllen und die E-Mail-Adresse zu bestätigen.

Als Belohnung für Ihre Bemühungen erhalten Sie per E-Mail den endgültigen Aktivierungsschlüssel.

11 Kontrollieren Sie Ihr E-Mail-Postfach. Es sollte eine Mail von Borland eingegangen sein. Die Mail enthält als Anlage eine Textdatei, die Sie unverändert in Ihrem Home-Verzeichnis abspeichern. (Unter Windows 2000/XP wäre dies das Verzeichnis *C:\Dokumente und Einstellungen\<Ihr_Benutzername>*[3].)

C++BuilderX das erste Mal starten

Der C++BuilderX ist ein Compiler mit einer integrierten Entwicklungsumgebung (kurz IDE für »Integrated Developing Environment«). Für uns als Programmierer bedeutet 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 (MS-DOS-Eingabeaufforderung) ausgeführt werden (siehe den im Anschluss beschriebenen Borland- oder den GNU-Compiler). Die ***integrierte Entwicklungsumgebung*** *des C++BuilderX ist eine Art Über-Programm, von dem aus alle für die Programmentwicklung benötigten Programme aufgerufen werden können. Sie verfügt über einen integrierten Editor, Dialogfenster und Menübefehle zum Aufruf des Compilers und eine ausgereifte Projektverwaltung.*

3 *Gemeint ist hier der Benutzername, unter dem Sie auf dem Rechner eingeloggt sind.*

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

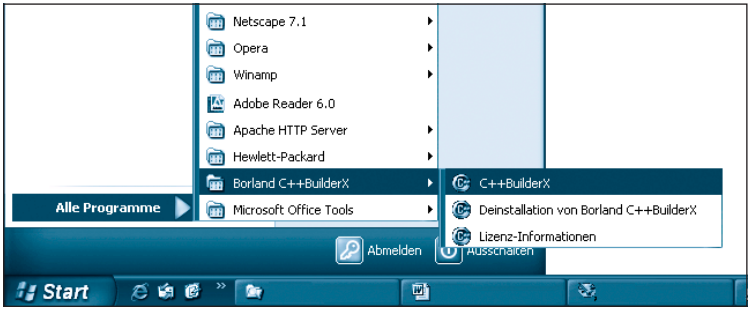


Abbildung 3.8: Aufruf des C++BuilderX über das Start-Menü

1 Doppelklicken Sie auf dem Desktop auf das C++BuilderX-Symbol oder klicken Sie in der Taskleiste auf die Schaltfläche *Start* und rufen Sie den Eintrag des C++BuilderX-Compilers aus der Borland-C++BuilderX-Programmgruppe auf.

Nach kurzer Ladezeit erscheinen die integrierte Entwicklungsumgebung des C++BuilderX und ein Dialogfenster, in dem Sie festlegen können, welche Dateiendungen mit dem C++BuilderX verbunden werden sollen (sodass der C++BuilderX automatisch gestartet wird, wenn Sie eine Datei mit einer dieser Endungen öffnen).

> **Hinweis**
>
> *Falls statt des C++BuilderX das Lizensierungsfenster erscheint, konnte der C++BuilderX die Aktivierungsdatei, die Ihnen per E-Mail zugesandt wurde, nicht auf der Festplatte finden. Wählen Sie im Lizenzierungsfenster die Option für den Aktivierungsschlüssel und geben Sie im nachfolgenden Dialog den Pfad zu Ihrer Aktivierungsdatei an. (Alternativ können Sie auch noch einmal kontrollieren, ob Sie die Aktivierungsdatei korrekt in Ihrem Home-Verzeichnis gespeichert habe, siehe letzter Schritt im vorherigen Abschnitt. Schieben Sie die Datei gegebenenfalls an die richtige Stelle und starten Sie den C++BuilderX neu.)*

2 Wenn Sie mit mehreren Compilern arbeiten möchten, registrieren Sie nur die Dateiendungen für die Projekte und Projektgruppen. Ansonsten registrieren Sie alle vorgeschlagenen Dateiendungen.

Nach Abschicken des Dialogs wird die Sicht auf den C++BuilderX frei.

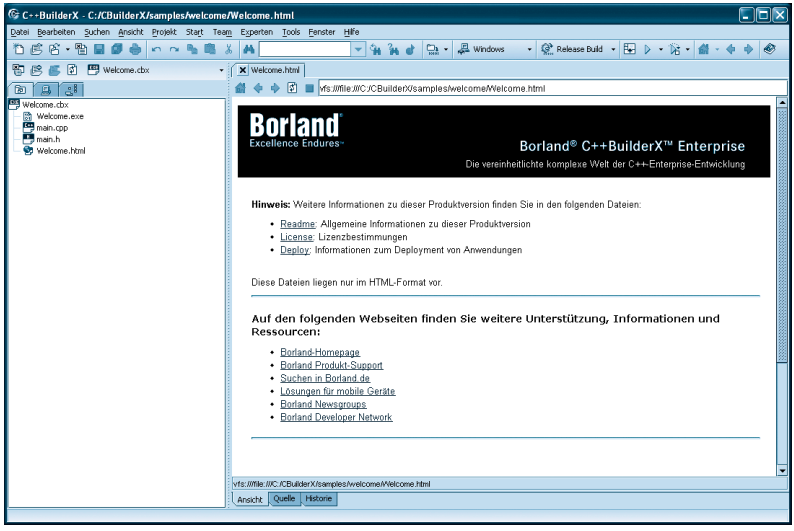


Abbildung 3.9: Die integrierte Entwicklungsumgebung des C++BuilderX

Neben der Menüleiste, mit den vielversprechend klingenden Menüeinträgen *Projekt* und *Start*, sehen Sie links das Projektfenster und rechts das Inhaltsfenster. Der C++BuilderX ist werksmäßig nämlich so eingestellt, dass er bei Programmstart automatisch das vorinstallierte Begrüßungsprojekt *Welcome.cbx* startet.

Für uns ist das Begrüßungsprojekt aber nicht weiter von Interesse; wir ziehen es vor, mit eigenen Projekten zu arbeiten.

3 Schließen Sie das Begrüßungsprojekt. Rufen Sie dazu den Befehl *Datei/Projekte schließen* auf, markieren Sie im gleichnamigen Dialog das Kästchen zu dem *Welcome.cbx*-Projekt und drücken Sie auf *OK*.

Ein neues Projekt anlegen

Im C++BuilderX werden Programme in Form von Projekten verwaltet (die wiederum in Projektgruppen organisiert werden).

Was ist das?

Die Quelldateien und Informationen, die zur Erstellung eines Programms benötigt werden, verwaltet der C++BuilderX 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 dem C++BuilderX besteht daher darin, ein passendes Projekt anzulegen.

1 Um ein neues Projekt anzulegen, rufen Sie den Menübefehl *Datei/Neu* auf.

Auf dem Bildschirm erscheint das Dialogfenster der Objektgalerie.

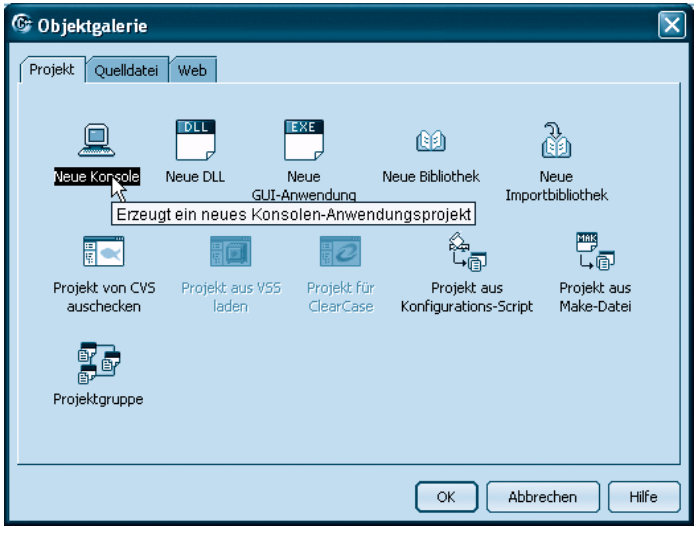


Abbildung 3.10: Die Objektgalerie des C++BuilderX

2 In der Objektgalerie sollte die *Projekt*-Seite angezeigt werden. Falls nicht, klicken Sie einfach auf das gleichnamige Register.

Auf der Seite *Projekt* wählen Sie aus, welche Art von Programm oder Programmmodul Sie erstellen und bearbeiten möchten. Die Auswahl ist recht umfangreich. Für uns ist allerdings nur ein Projekttyp interessant: *Neue Konsole*, der ein neues Projekt für eine Konsolenanwendung erstellt.

Hinweis

Konsolenanwendungen haben keine grafische Benutzeroberfläche, ja im Grunde genommen verfügen sie über gar keine eigene Benutzeroberfläche. Stattdessen nutzen sie zur Ein- und Ausgabe die Konsole, von der sie aufgerufen wurden (unter Windows ist dies die (MS-DOS-Eingabeaufforderung). Konsolenanwendungen haben aber den Vorteil, dass man sich ganz auf den reinen C++-Code konzentrieren kann. Im Übrigen werden Konsolenanwendungen auch heute noch eingesetzt – beispielsweise als Hilfsprogramme für Webserver. Und wenn Sie später zur Window-Programmierung überwechseln wollen ... die reine C++-Programmierung bleibt stets die gleiche, Sie müssen nur noch lernen, wie man Programme Window-fähig macht.

3 Wählen Sie das *Neue Konsole*-Symbol aus und klicken Sie auf *OK*.

Es erscheint das Dialogfeld des Konsolenanwendungsexperten, der Ihnen bei der weiteren Konfiguration des neu anzulegenden Projekts hilft. Wichtig sind dabei vor allem die Einstellungen im ersten Dialog des Experten.

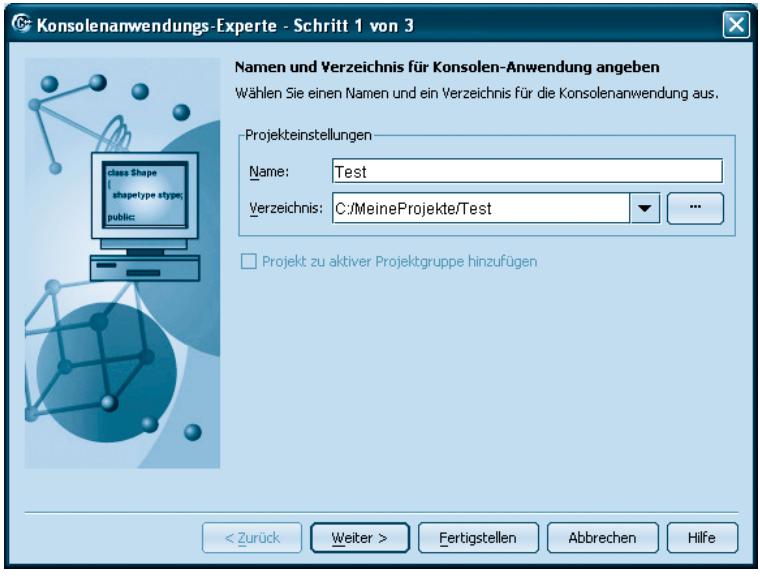


Abbildung 3.11: Name und Verzeichnis des Projekts festlegen

4 Im ersten Dialog des Experten legen Sie Namen und Verzeichnis für das Projekt fest. Danach klicken Sie auf *Weiter*.

Aufgepasst! Um von Anfang an Ordnung auf Ihrer Festplatte zu halten, sollten Sie auf Ihrer Festplatte ein eigenes Verzeichnis anlegen, unter dem Sie alle C++-Projekte speichern, beispielsweise *C:\MeineProjekte*. Neue Projekte speichern Sie dann unter diesem Verzeichnis in eigenen Unterverzeichnissen, die den gleichen Namen wie die Projekte tragen.

Tippen Sie nun das übergeordnete Verzeichnis (beispielsweise also *C:\MeineProjekte*) in das Eingabefeld *Verzeichnis* ein. Wenn Das Verzeichnis bereits existiert, können Sie es über den …-Schalter auswählen. Anschließend hängen Sie an den Verzeichnispfad noch einen zusätzlichen Schrägstrich / an.

Wechseln Sie in das Eingabefeld *Name* und geben Sie dort den gewünschten Projektnamen ein (der im Übrigen auch der Name des Programms sein wird). Der Assistent sollte dabei den Verzeichnispfad automatisch um ein gleichnamiges Unterverzeichnis erweitern.

Achtung

Name und Verzeichnis dürfen kein +-Zeichen enthalten!

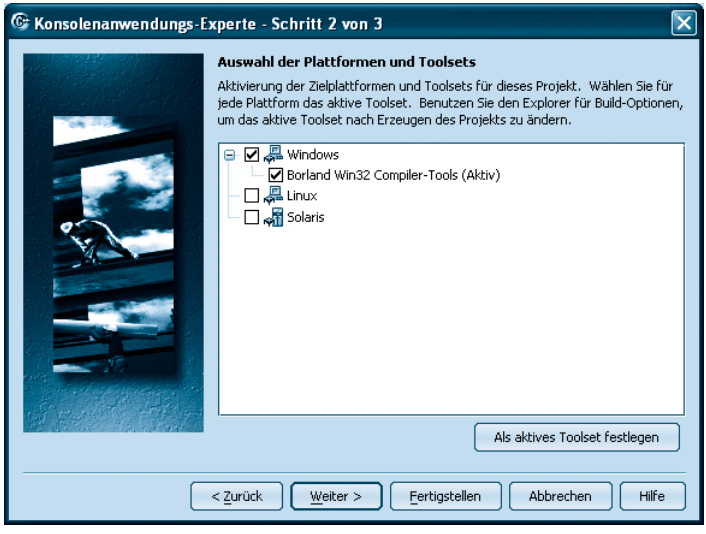


Abbildung 3.12: Die Zielplattform auswählen

5 Vergewissern Sie sich, dass im zweiten Dialog des Assistenten die Option *Windows* und darunter die Option *Borland Win32 Compiler-Tools* aktiviert sind.

Hinweis

Sollte die zweite Option gar nicht angezeigt werden, ist etwas bei der Installation schief gelaufen. In diesem Fall sollten Sie den C++-BuilderX zuerst de- und dann neu installieren (wobei Sie besonders auf die Einstellungen im Dialog Produktkomponenten auswählen *achten).*

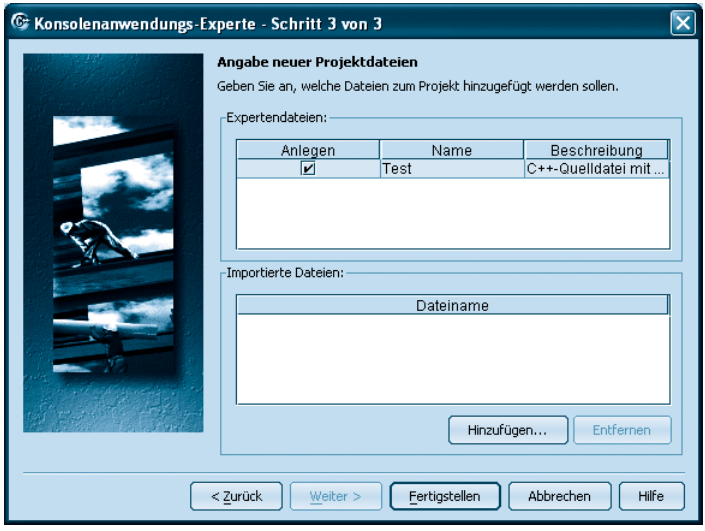


Abbildung 3.13: Projekt mit einer ersten Quelltextdatei starten

6 Wenn Sie möchten, können Sie im letzten Dialog des Konsolenexperten gleich auch eine Quelltextdatei anlegen lassen. Doppelklicken Sie auf das Feld unter *Name* und ersetzen Sie *untitled* durch den Projektnamen. Markieren Sie dann das Kästchen unter der Spalte *Anlegen* und klicken Sie anschließend auf *Fertigstellen*.

Hinweis

Später, wenn Ihre Projekte mehrere Dateien umfassen, sollten Sie den einzelnen Quelldateien Namen geben, die auf ihre Bedeutung für das Programm oder ihren Inhalt schließen lassen. Für den Anfang, wo alle unsere Programme aus einer einzigen Quelldatei bestehen, ist es am übersichtlichsten, wenn Sie die Quelldatei des Projekts genauso nennen wie das Projekt selbst.

Der C++BuilderX legt jetzt das neue Projekt an. Es verfügt über eine Quelldatei namens *Test.cpp*, die der C++BuilderX bereits mit einem rudimentären Codegerüst versehen hat. Um sich das Codegerüst anzuschauen, brauchen Sie nur im Projektfenster auf den Eintrag für die Quelldatei doppelt zu klicken.

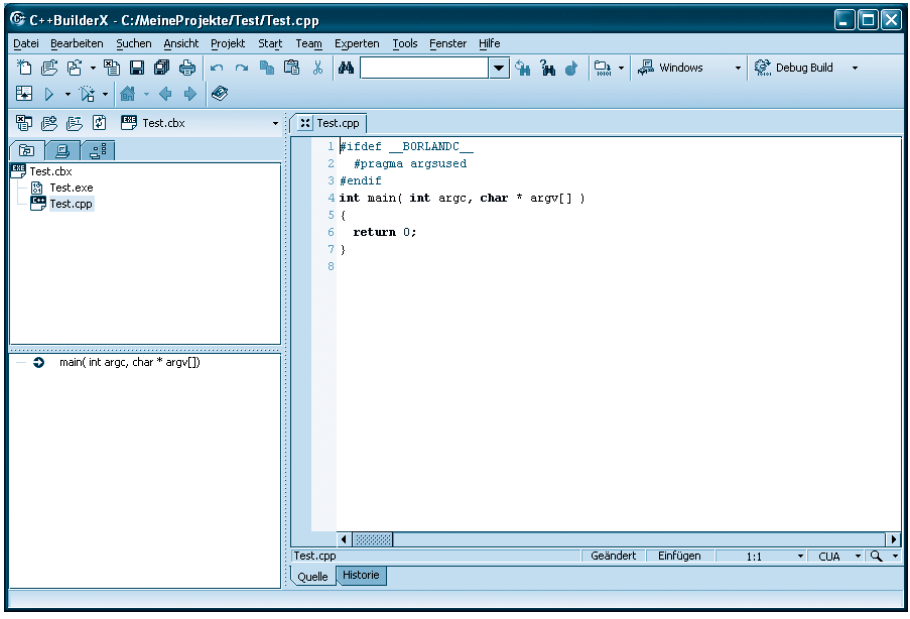


Abbildung 3.14: Die IDE nach dem Anlegen des Projekts

Wenn Sie später einmal größere Projekte angehen, die aus mehreren Dateien bestehen, siehe letztes Kapitel, können Sie dem Projekt über den Befehl *Datei/Neue Datei* oder über *Datei/Neu*, Seite *Quelldatei* zusätzliche Quelldateien hinzufügen.

Hinweis

Die weiteren Beispielprogramme aus diesem Buch sind nach Kapiteln geordnet. Die Programme jedes Kapitels stehen in einem eigenen Unterverzeichnis: Kap04, Kap05, Kap06 *und so weiter. Jedes Programm besitzt ein eigenes Projektverzeichnis (*Kap06/Ersetzen, Kap06/Fahrenheit*). Die Projektdateien tragen den gleichen Namen wie die Projektverzeichnisse.*

Den Quelltext aufsetzen

Nachdem die Quelldatei des Projekts in den Editor geladen wurde, können Sie den Quelltext des Programms eingeben.

> **Tipp**
>
> *Wenn Sie einmal nicht sicher sind, ob das richtige Projekt und die richtige Datei geladen wurden, schauen Sie einfach in die Titelleiste der C++BuilderX-IDE. Dort werden Pfad und Name der gerade geladenen Datei angezeigt. Den einfachen Namen der Quelldatei können Sie zudem auf dem zugehörigen Reiter am oberen Rand des Editorfensters lesen. Wenn Sie mehrere Dateien in den Editor geladen haben, können Sie die Dateien über die Reiter auswählen.*

1 Tippen Sie den folgenden Programmquelltext ein.

Wenn Sie möchten, können Sie das vom C++BuilderX angelegte Programmgerüst anpassen. Einfacher und übersichtlicher dürfte es jedoch sein, das vorgegebene Programmgerüst ganz zu löschen und den Quelltext komplett abzutippen.

```
// 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 *Test.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!*

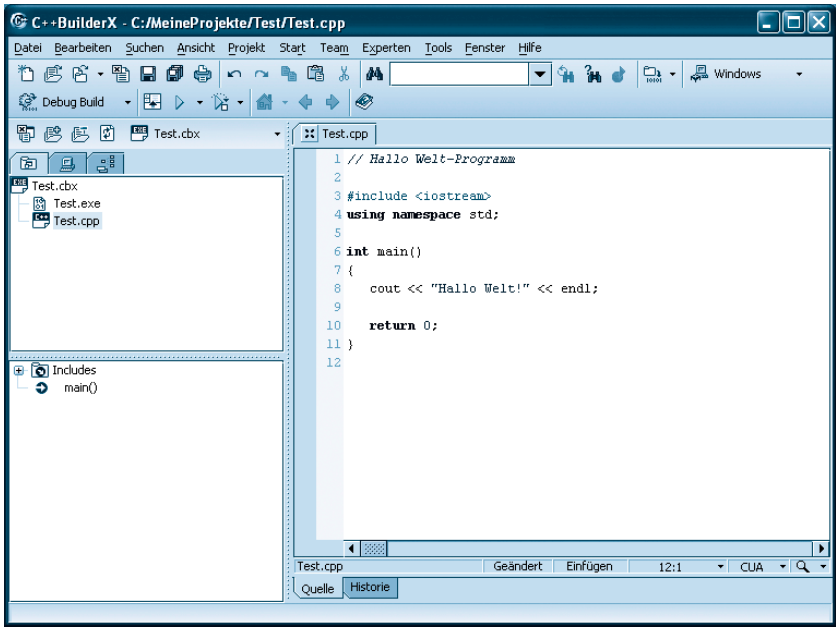


Abbildung 3.15: Der Programmcode wurde in die Datei Test.cpp eingetippt

Den Quelltext kompilieren

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 C++BuilderX-Entwicklungsumgebung beim Kompilieren (oder späteren Ausführen) des Programms abstürzen sollte.

1 Speichern Sie die Quelldatei durch Aufruf des Befehls *Datei/"Test.cpp" speichern* oder durch Drücken der Tastenkombination Strg+S.

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.

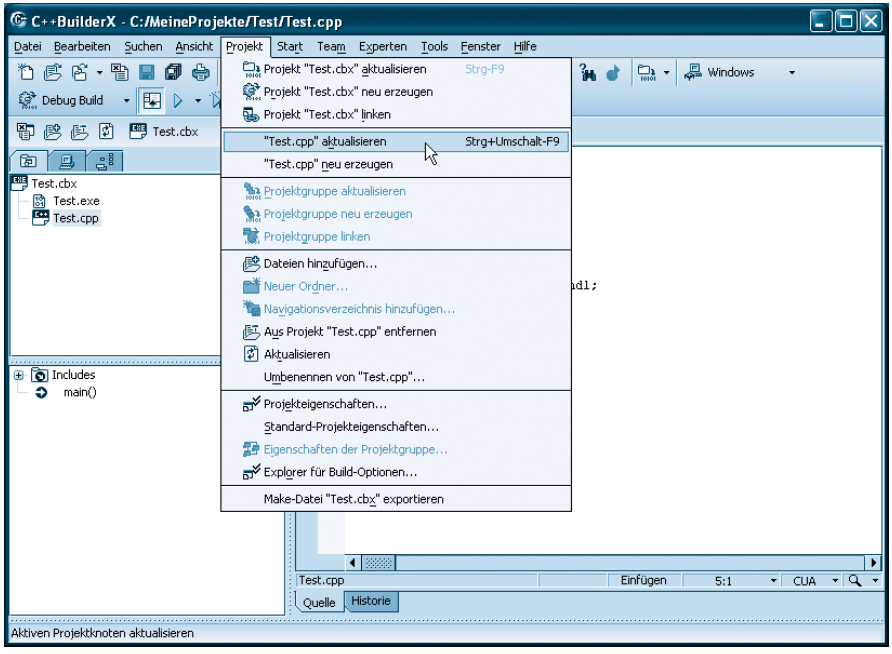


Abbildung 3.16: Die aktuelle Quelltextdatei kompilieren

2 Rufen Sie den Befehl *Projekt/"Test.cpp" aktualisieren* auf, um den Quelltext zu kompilieren.

Für einen kurzen Moment erscheint das Dialogfenster *Fortschritt*, welches Sie über den Fortgang der Kompilation informiert. Gleichzeitig wird am unteren Rand der IDE das Meldungsfenster mit dem Reiter *Compiler* eingeblendet. Im Idealfall sollte dort der Name der kompilierten Quelldatei, ohne Angabe von Fehlern und Warnungen, zu lesen sein.

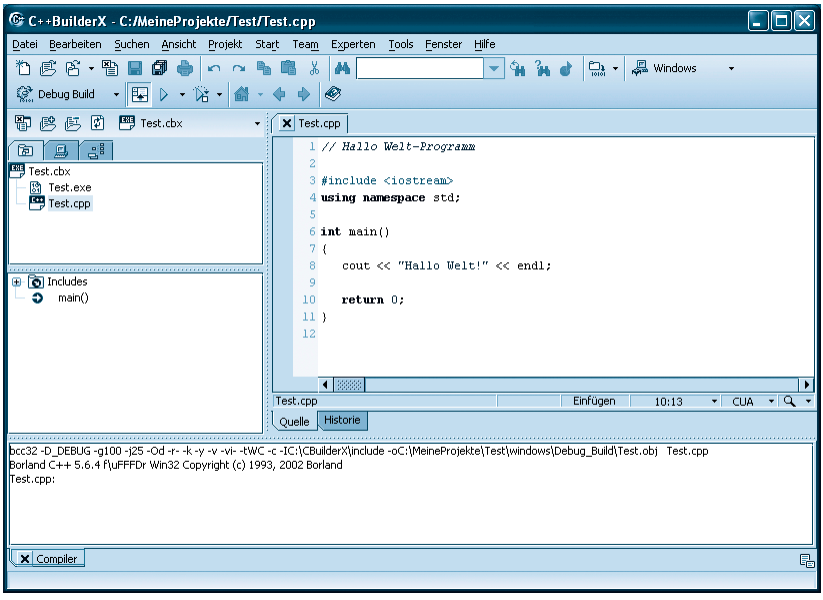


Abbildung 3.17: Keine Fehler – super!!!

Es kommt jedoch 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 so genannten »syntaktischen Fehler« noch recht harmlos, denn sie werden – im Gegensatz zu den logischen Fehlern – vom Compiler entdeckt und meist recht gut lokalisiert.

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 *Projekt/"Test.cpp" aktualisieren* auf.

Das Meldungsfenster weist Sie nun daraufhin, dass ein Fehler aufgetreten ist.

> **Tipp**
>
> *Schneller kompilieren Sie durch Drücken der Tastenkombination* Strg + ⇧ + F9.

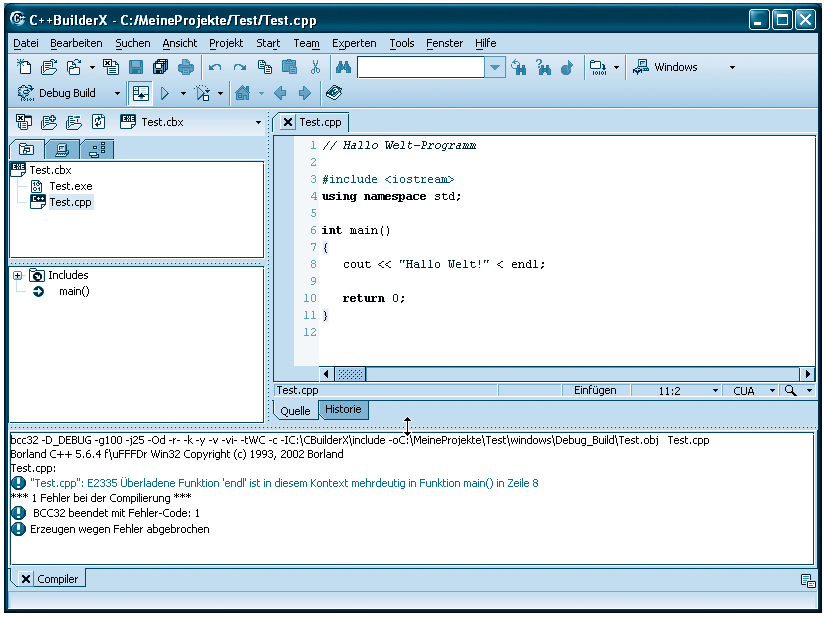


Abbildung 3.18: Beim Kompilieren wurde ein Fehler festgestellt

In unserem Fall gibt es nur eine einzige Fehlermeldung (rote Zeile) Die Quelltextzeile, die nach Einschätzung des Compilers für den Fehler verantwortlich ist, wird am Ende der Fehlermeldung angegeben: hier Zeile 8.

Wenn Sie im Meldungsfensters auf die Fehlermeldung doppelklicken, wird die auslösende Quelltextzeile im Editor rot unterlegt. Wenn Sie die Fehlermeldung markieren und F1 drücken, wird die Online-Hilfe geöffnet und Sie erhalten ausführlichere Erklärungen zu dem aufgetretenen Fehler. Auf diese Weise hilft Ihnen der Compiler dabei, syntaktisch korrekte Quelltexte aufzusetzen!

Beachten Sie aber, dass der Compiler die Fehler nicht immer korrekt einschätzt und lokalisiert (sonst könnte er sie ja gleich an Ihrer statt korrigieren). Die in unserem Fall erzeugte Fehlermeldung ist beispielsweise eher irreführend als hilfreich, da sie auf `endl` statt auf den fehlerhaften Operator < hinweist.

Tipp

Wenn mehrere Fehlermeldungen erzeugt werden, sollten Sie diese von oben nach unten abarbeiten und zwischendurch mehrmals neu kompilieren, denn es ist gut möglich, dass es sich bei den unteren Fehlermeldungen um Nachfolgefehler handelt, die automatisch verschwinden, wenn der vorangehende Fehler behoben wurde.

Warnungen sind nicht ganz so schwerwiegend wie Fehler. Sie werden ausgegeben, wenn der Compiler auf syntaktisch korrekten Quelltext trifft, den er aber trotzdem für logisch falsch hält. Prüfen Sie auf jeden Fall, ob der Compiler die Warnung zu Recht ausgegeben hat, oder ob der betreffende Code unkritisch ist.

3 Korrigieren Sie den Fehler.

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

4 Drücken Sie Strg+⇧+F9, um das Programm erneut kompilieren zu lassen.

Das Programm erstellen und testen

Treten beim Kompilieren keine Fehler auf, übersetzt der Compiler den Quelltext in Maschinencode und speichert diesen in einer eigenen Datei mit der Extension *.obj*. Damit haben wir aber noch kein ausführbares Programm. Dazu müssen wir erst noch den Linker aufrufen, der aus dem Maschinencode in der obj-Datei eine ausführbare exe-Datei macht.

1 Rufen Sie den Befehl *Projekt/Projekt "Test.cbx" aktualisieren* auf. Fortschritt und Erfolg des Linkvorgangs können Sie wiederum im Fortschrittsdialog und im Meldungsfenster kontrollieren. Wieder gilt: keine Meldungen sind gute Meldungen.

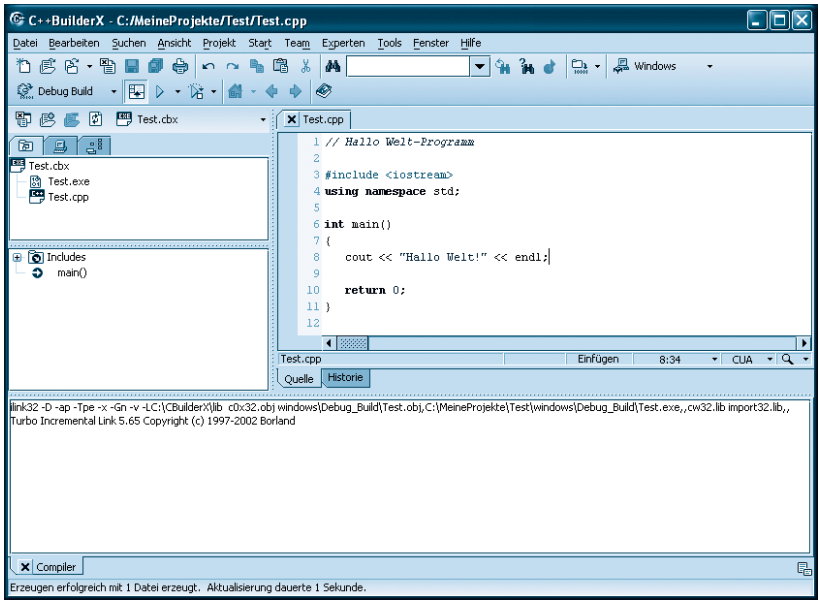


Abbildung 3.19: Das Programm wurde erfolgreich gelinkt, die EXE-Datei erzeugt

Tipp

Der Befehl Projekt "Test.cbx" aktualisieren *kann auch zum Kompilieren verwendet werden. Enthält das Projekt Quelldateien, die noch nicht kompiliert oder seit der letzten Kompilierung überarbeitet wurden, ruft der Befehl zuerst den Compiler und dann den Linker auf. Wenn Sie also nach Bearbeitung des Quelltextes das Programm erstellen wollen, brauchen Sie nicht zuerst den Befehl* "Quelldatei" aktualisieren *aufrufen, sondern können gleich den* Projekt "Projektname" aktualisieren *-Befehl ausführen.*

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.

2 Rufen Sie den Befehl *Start/Projekt ausführen* auf.

Der C++BuilderX lädt das Programm und führt es aus. Als »Konsole« muss wiederum das Meldungsfenster herhalten. Hier sollten Sie die Ausgabe des Programms

```
Hallo Welt!
```

lesen können.

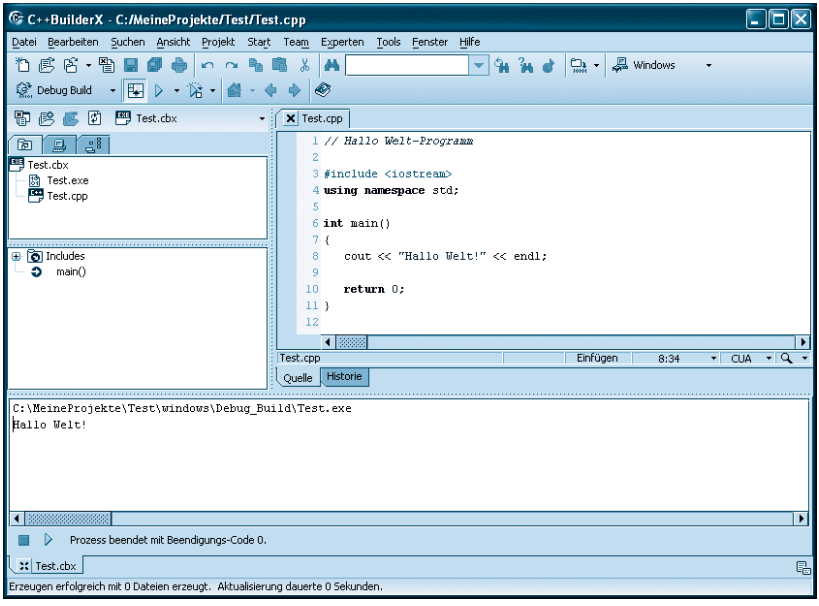


Abbildung 3.20: Das Programm arbeitet wie erwartet

Das Programm ausführen

Natürlich will man nicht immer den C++BuilderX aufrufen, um ein selbst erstelltes Programm auszuführen. Aber das ist ja auch nicht nötig. Die EXE-Datei des Programms ist auf der Festplatte abgespeichert und kann direkt aufgerufen werden.

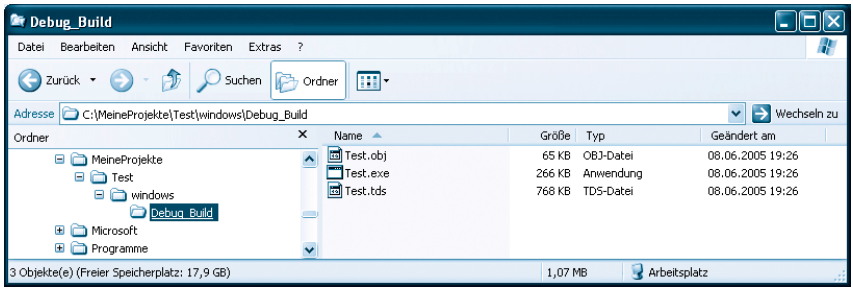


Abbildung 3.21: Das Projekt auf der Festplatte

1 Öffnen Sie im Windows Explorer das Verzeichnis des Projekts und expandieren Sie die Unterverzeichnisse *windows* und *Debug_Build*.

> **Hinweis**
>
> *Erschrecken Sie nicht wegen der zusätzlichen Dateien im Verzeichnis Ihres Projekts* (Test.tds, Test.obj) *Dies sind lediglich Hilfsdateien, die vom C++BuilderX automatisch angelegt wurden. Wenn das Programm fertig und ausgetestet ist, können Sie alle Dateien mit den Endungen tds und obj sowie eventuell angelegte Sicherungsdateien (zu erkennen an der Tilde ~ in der Dateierweiterung) löschen.*

2 Doppelklicken Sie auf die EXE-Datei.

Das Konsolenprogramm wird daraufhin ausgeführt. Da das Programm über kein eigenes Fenster verfügt, richtet Windows für das Programm ein Konsolenfenster ein. Leider schließt Windows das Konsolenfenster auch direkt wieder, wenn das Programm beendet ist. Im Falle unseres Test-Programms bleibt Ihnen dadurch kaum Zeit die Ausgabe des Programms zu kontrollieren, ja womöglich werden Sie das Öffnen und Schließen des Konsolenfenster nur als kurzes Aufblinken wahrnehmen!

In diesem Fall können Sie auf zweierlei Wegen Abhilfe schaffen.

Erstens: Sie rufen am Ende des Programms die Methode `cin.get()` auf 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.

3 Rufen Sie die MS-DOS-Eingabeaufforderung auf. Öffnen Sie dazu das *Start*-Menü von Windows und wählen Sie je nach Betriebssystem den Eintrag *Programme/MS-DOS-Eingabeaufforderung* oder *Programme/Zubehör/Eingabeaufforderung* auf. (Unter manchen Windows-Versionen können Sie die Eingabeaufforderung aufrufen, indem Sie im Dialog des Befehls *Start/Ausführen* `cmd` eingeben und abschicken.)

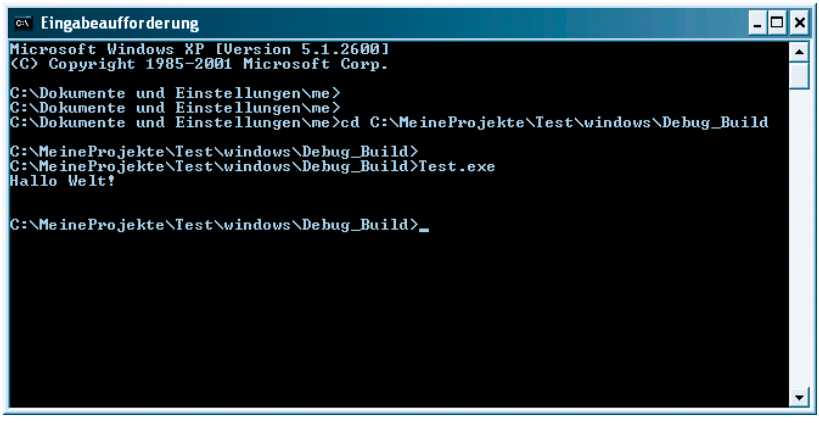


Abbildung 3.22: Ausführung in einem Konsolenfenster

4 Wechseln Sie mit Hilfe des `cd`-Befehls in das Verzeichnis, in der die EXE-Datei steht.

5 Rufen Sie das Programm von der Konsole aus auf, indem Sie hinter dem Prompt den Namen der EXE-Datei eintippen und mit der ↵-Taste abschicken.

Wenn Sie statt der Ausgabe des Programms eine Fehlermeldung erhalten, dass der »Befehl« womöglich falsch geschrieben oder nicht gefunden wurde, kann dies daran liegen, dass Sie sich im falschen Verzeichnis befinden oder Sie vergessen haben, die EXE-Datei zu erstellen. Kontrollieren Sie dann zur Sicherheit, welche Dateien sich im aktuellen Verzeichnis befinden, indem Sie von der Konsole aus den Befehl `dir` oder `dir/p` abschicken.

> **Hinweis**
>
> *Ausführlichere Informationen zur Windows-Eingabeaufforderung finden Sie in dem Tutorial »Arbeiten mit der Windows-Konsole«, das Sie von der Website* `www.carpelibrum.de` *herunterladen können.*

Projekte schließen und C++BuilderX beenden

Wenn Sie mit der Arbeit an Ihrem Programm fertig sind oder ein neues Projekt beginnen wollen, speichern Sie Ihre Arbeit und schließen Sie dann das aktuelle Projekt.

1 Rufen Sie den Befehl *Datei/Alles Speichern* auf.

2 Schließen Sie das Projekt über den Befehl *Datei/Projekte schließen*.

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

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

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

- Sie können den Befehl *Datei/Projekt öffnen* aufrufen und die cbx-Datei aus dem Projektverzeichnis laden. (So können Sie auch die Projekte auf der Buch-CD öffnen.)
- Kürzlich bearbeitete Projekte können Sie über den Befehl *Datei/Neu öffnen* laden.

Kompilierung der Beispielprogramme auf der CD

Die Beispiele zu diesem Buch wurden alle mit dem C++BuilderX erstellt und stehen auf der Buch-CD. Wenn Sie eines der Beispiele mit dem C++BuilderX kompilieren möchten, kopieren Sie einfach das betreffende Projektverzeichnis von der Buch-CD auf Ihre Festplatte, laden Sie die Projektdatei (erkenn-

bar an der Extension .cbx) in den C++BuilderX, kompilieren Sie das Projekt und führen Sie das Programm aus (siehe auch Anhang D).

> **Achtung**
>
> *Linux-Anwender müssen die Projektdateien vor dem Kompilieren neu konfigurieren. Rufen Sie dazu nach dem Laden des Projekts den Menübefehl* Projekt/Projekteigenschaften *auf und gehen Sie zur Seite* Plattformen*. Deaktivieren Sie dort den Windows-Knoten und aktivieren Sie den Linux-Knoten mit dem GNU-C++-Compiler.*

Programmerstellung mit dem reinen Borland 5.5-Compiler (Windows)

Nicht alle Compiler werden mit integrierter Entwicklungsumgebung und Projektverwaltung ausgeliefert. Viele gute C++-Compiler sind reine Kommandozeilenwerkzeuge, die von der Konsole aus aufgerufen werden. Der Leistungsfähigkeit dieser Compiler tut dies keinen Abbruch, es fehlt halt nur die Annehmlichkeit der integrierten, grafischen Benutzeroberfläche.

Als Beispiel für einen Kommandozeilen-Compiler möchte ich Ihnen die Kommandozeilenversion des Borland 5.5-Compilers vorstellen, den Sie kostenlos von der Borland-Website `http://www.borland.de` herunterladen können.

Installation

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

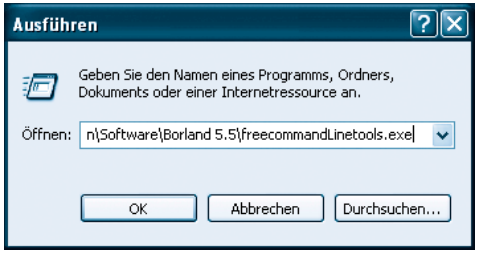


Abbildung 3.23: Start des Installationsprogramms über Start/Ausführen.

2 Führen Sie die EXE-Datei des Installationsprogramms aus, um den Compiler zu installieren. Öffnen Sie beispielsweise das Start-Menü und wählen den Befehl *Ausführen* aus 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. Installieren Sie den Compiler am besten in das vorgeschlagene Verzeichnis *C:\Borland\BCC55*.

Nach der Installation sollten Sie Ihr System so anpassen, dass Sie den Compiler möglichst bequem aufrufen können.

3 Tragen Sie dazu das BIN-Verzeichnis des Compilers in Ihren Systempfad ein:

Wenn Ihr Windows-Betriebssystem die *C:\Autoexec.bat*-Datei auswertet, können Sie diese in einen Texteditor laden und am Ende der Datei folgende Zeile hinzufügen:

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

Ansonsten rufen Sie den zugehörigen Windows-Konfigurationsdialog auf und erweitern Sie den Wert der PATH-Variable um den Eintrag `C:\Borland\Bcc55\bin;`.

Wie Sie den Konfigurationsdialog aufrufen, hängt von dem jeweiligen Windows-Betriebssystem ab. In einigen Versionen können Sie den Dialog aufrufen, indem Sie im *Start*-Menü den Befehl *Ausführen* auswählen und in dem erscheinenden Eingabefeld `msconfig` eintippen. In den Versionen der NT/2000/XP-Familie klicken Sie in der Systemsteuerung (klassische Ansicht) auf das Symbol *System*, wechseln zur Seite *Erweitert* und klicken dort auf die Schaltfläche *Umgebungsvariablen*. Im Listenfeld für die Systemvariablen wählen Sie dann die Variable `Path` aus und drücken den *Bearbeiten*-Schalter.

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

4 Damit der Borland-Compiler seine Bibliotheken findet, müssen Sie im *Bin*-Verzeichnis des Compilers zwei Textdateien *bcc32.cfg* und *ilink32.cfg* anlegen.

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 *Borland55* 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.*

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

Um zu prüfen, ob der Compiler korrekt installiert und eingerichtet ist, rufen Sie nach dem Neustart die Windows-Konsole auf (*Start/Programme/MS-DOS-Eingabeaufforderung* oder *Start/Programme/Zubehör/Eingabeaufforderung*) und tippen Sie `bcc32` ein. In der Konsole sollte daraufhin eine Meldung des Compilers erscheinen.

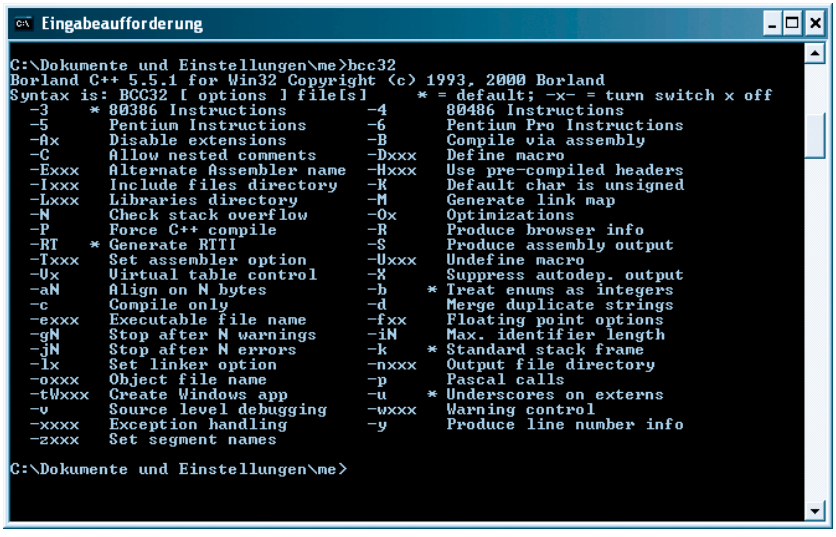


Abbildung 3.24: Der Compiler ist korrekt installiert.

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

Ist der Compiler ordnungsgemäß installiert und eingerichtet, können Sie Ihre Programme wie folgt kompilieren:

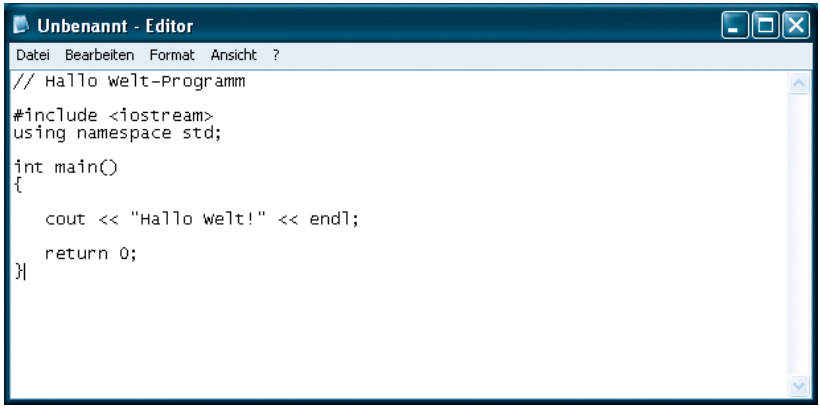


Abbildung 3.25: Zum Eintippen der Quelltexte genügt ein einfacher Editor.

1 Tippen Sie den Quelltext des Programms in einen Texteditor ein – beispielsweise Notepad (Aufruf über *Start/Programme/Zubehör/Editor* oder im *Start/Ausführen*-Dialog notepad eingeben).

2 Speichern Sie das Programm mit der Extension .cpp.

Um von Anfang an Ordnung auf Ihrer Festplatte zu halten, sollten Sie auf Ihrer Festplatte ein eigenes Verzeichnis für Ihre zukünftigen C++-Projekte anlegen, beispielsweise *C:\MeineProjekte*. Für jedes neue Programm, das Sie schreiben möchten, legen Sie dann unter diesem Verzeichnis ein eigenes Unterverzeichnis an, das den gleichen Namen wie das Programm trägt. In diesem Unterverzeichnis speichern Sie die Quelltextdateien des Programms. Gibt es nur eine Quelltextdatei nennen Sie diese einfach ebenso wie das Programm.

Zum Testen der Programmerstellung schlage ich Ihnen vor, unter *C:\MeineProjekte* einfach ein Unterverzeichnis *Test* anzulegen und darunter die in Schritt 1 aufgesetzte Quelltextdatei als *Test.cpp* zu speichern.

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 der* Öffnen mit*-Dialog. Wählen Sie als gewünschtes Bearbeitungsprogramm Notepad aus und aktivieren Sie die Option* Diese Datei immer mit diesem Programm öffnen*. Wenn Sie den Dialog jetzt abschicken, wird die Extension .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 angehängte .txt-Extensionen haben..*

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

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

5 Kompilieren Sie die Datei mit folgendem Befehl:

```
bcc32 Test.cpp
```

6 Führen Sie das Programm in der Konsole aus. (Namen der EXE-Datei eintippen und abschicken).

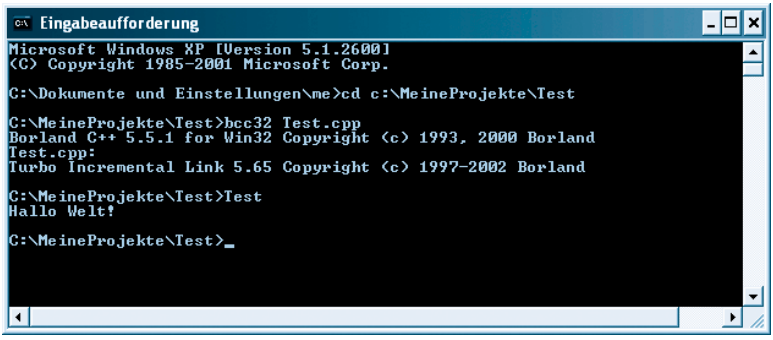


Abbildung 3.26: Ausführung in einem Konsolenfenster

Kompilierung der Beispielprogramme auf der CD

Die Beispiele auf der Buch-CD wurden alle mit dem C++BuilderX erstellt und sind daher in Projekte eingebettet. Wenn Sie eines der Beispiele mit dem 5.5-Compiler kompilieren möchten, kopieren Sie einfach das betreffende Projektverzeichnis von der Buch-CD auf Ihre Festplatte, wechseln Sie in das kopierte Projektverzeichnis und kompilieren Sie die CPP-Datei des Projekts (siehe auch Anhang D).

> **Hinweis**
>
> *Die Erstellung von Programmen aus mehreren Quelldateien ist in Kapitel 16 beschrieben.*

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

Als Beispiel für einen Linux-Compiler sei hier die Programmerstellung mit dem g++-Compiler 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 `www.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 3.27: 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 3.28: 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.

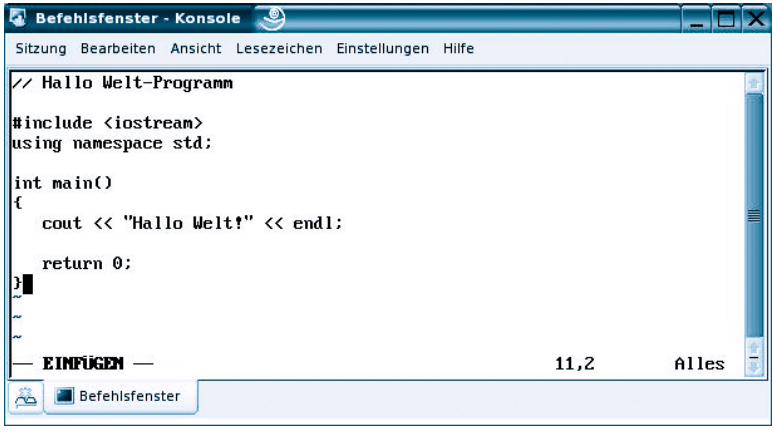


Abbildung 3.29: Quelltext eintippen

4 Geben Sie den Programmquelltext ein.

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

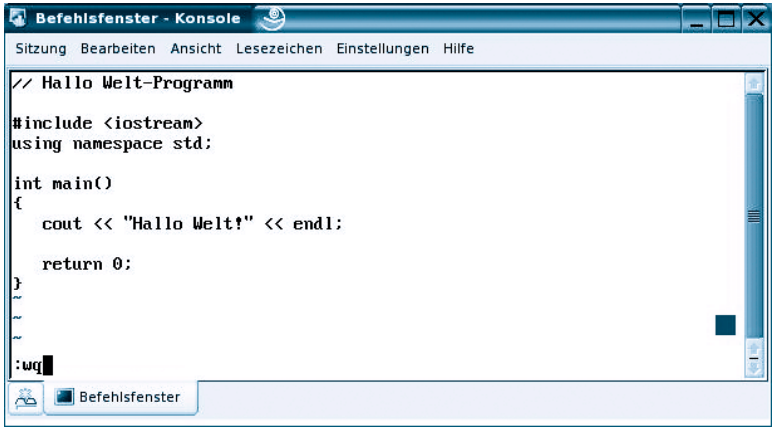


Abbildung 3.30: Speichern und beenden

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

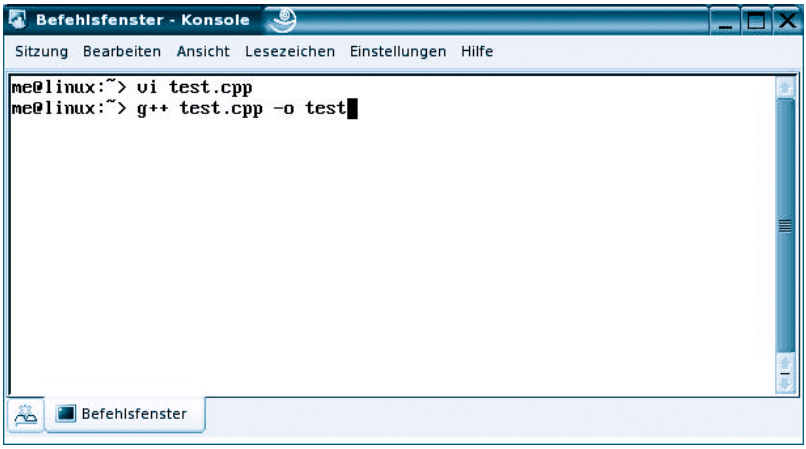


Abbildung 3.31: Kompilieren mit dem g++-Compiler

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.

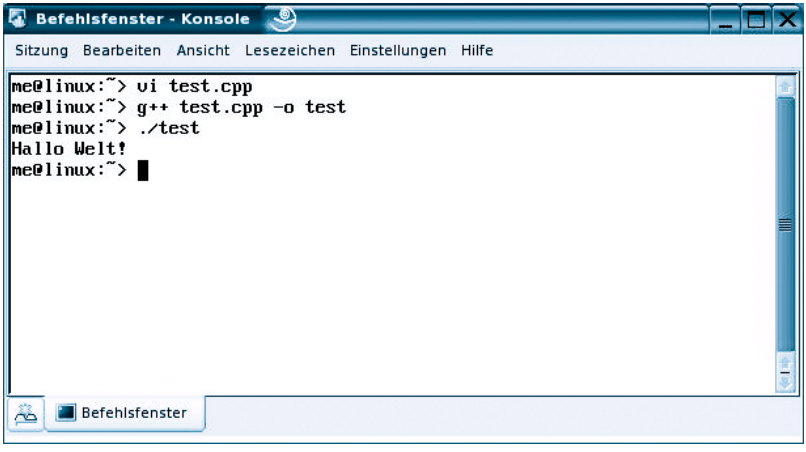


Abbildung 3.32: Ausführung in einem Konsolenfenster

8 Führen Sie das Programm aus.

Kompilierung der Beispielprogramme auf der CD

Die Beispiele auf der Buch-CD wurden alle mit dem C++BuilderX erstellt und sind daher in Projekte eingebettet. Wenn Sie eines der Beispiele mit dem g++-Compiler kompilieren möchten, kopieren Sie einfach das betreffende Projektverzeichnis von der Buch-CD auf Ihre Festplatte, wechseln Sie in das kopierte Projektverzeichnis und kompilieren Sie die CPP-Datei des Projekts (siehe auch Anhang D).

> **Hinweis**
>
> *Die Erstellung von Programmen aus mehreren Quelldateien ist im Anhang beschrieben.*

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.