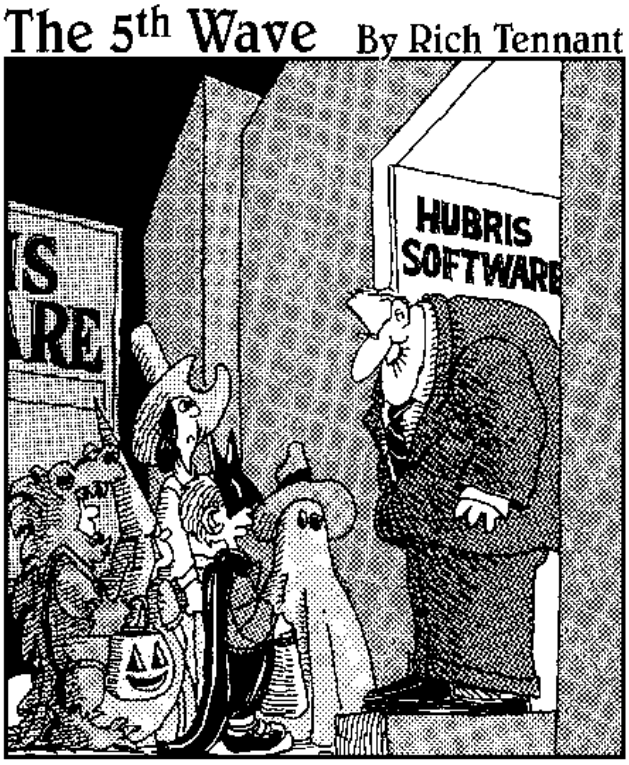
Teil I

Einführung in die C++-Programmierung



»Wir haben dieses Jahr unseren Starttermin für TREAT am 31. Oktober verpasst, aber wir entwickeln eine neue, robustere Version, die im ersten oder zweiten Quartal des nächsten Jahres verfügbar sein sollte.«

In diesem Teil ...

Sowohl der neueste, heißeste Flugsimulator als auch die einfachsten, doch leistungsstärksten Buchhaltungsprogramme verwenden dieselben Grundbausteine. In diesem Teil lernen Sie die grundlegenden Funktionen kennen, die Sie benötigen, um Ihre Killer-Anwendung zu schreiben.

Ihr erstes C++-Programm

1

In diesem Kapitel

- C++ kennen lernen
- Dev-C++ von der beiliegenden CD-ROM installieren
- Ihr erstes C++-Programm erstellen
- Ihr Programm ausführen

Okay, da sind wir: Nur Sie und ich, niemand sonst. Wir müssen nur noch anfangen. Warum fangen wir nicht mit einigen Grundbegriffen an?

Ein Computer ist eine erstaunlich schnelle, aber unglaublich dumme Maschine. Ein Computer kann (innerhalb vernünftiger Grenzen) alles tun, was Sie ihm sagen, aber er tut *genau* das, was Sie ihm sagen – nicht mehr und nicht weniger.

Leider verstehen Computer keine normale menschliche Sprache – weder Englisch noch Deutsch. Okay, möglicherweise wenden Sie ein, dass Sie Computer gesehen haben, die auf natürliche Sprache reagierten. Doch was Sie tatsächlich gesehen haben, war ein Computer, der ein *Programm* ausführte, das sinnvolle natürlichsprachliche Aussagen verstehen konnte. (Ich bin mir noch nicht ganz sicher, was es mit dieser Verarbeitung der natürlichen Sprache durch Computer auf sich hat, aber da ich auch nicht weiß, ob mein Sohn meine Ratschläge versteht, lasse ich es dabei bewenden.)

Computer verstehen eine Sprache, die als *Programmiersprache* oder *Maschinensprache* bezeichnet wird. Es ist für Menschen möglich, aber extrem schwierig, Maschinensprache zu sprechen. Deshalb wurden Sprachen wie beispielsweise C++ erfunden, mit denen sich Menschen und Computer gewissermaßen in der Mitte treffen und verständigen können. Menschen können C++ »sprechen« oder »schreiben«, und C++ wird in Maschinensprache übersetzt, die der Computer versteht.

C++-Begriffe verstehen

In den frühen 70er Jahren des vergangenen Jahrhunderts entwickelte eine Gruppe wirklich kluger Leute ein Computersystem namens *Multix*. Der Zweck von Multix bestand darin, allen Anwendern einen preiswerten Computerzugang zu Grafiken, E-Mail, Börsenkursen, Pornografie (kleiner Scherz) usw. zu ermöglichen. Natürlich war diese Idee damals vollkommen verrückt, und das gesamte Konzept scheiterte.

Ein kleines Team von Ingenieuren, die für die *Bell Labs* arbeiteten, beschloss, einen Teil von Multix in einem sehr kleinen, kompakten Betriebssystem zu bewahren, das als *Unix* bezeichnet wurde (*Un*-ix, die Single-Tasking-Version von *Mult*-ix, alles klar?).

Leider verfügten diese Ingenieure nicht über eine einzige große Maschine, sondern hatten eine Reihe kleinerer Rechner jeweils von einem anderem Hersteller. Damals waren die normalen Verfahren zur Entwicklung von Programmen alle maschinenabhängig; deshalb hätten sie dasselbe Programm für jede verfügbare Maschine neu schreiben müssen. Stattdessen erfanden diese Ingenieure eine kleine, leistungsstarke Sprache, die sie *C* nannten.

C verbreitete sich wie ein Lauffeuer. Im Laufe der Zeit wurden jedoch neue Programmiertechniken (insbesondere die objektorientierte Programmierung) erfunden, für die die Programmiersprache C nicht mehr geeignet war. Um die Entwicklung nicht zu verpassen, wurde die Sprache C um die einschlägigen neuen Konzepte erweitert. Das Ergebnis war eine Sprache, die als *C++* bezeichnet wurde.

Die C++-Sprache besteht aus zwei grundlegenden Elementen:

- **Semantik:** Dies ist ein Vokabular von Befehlen, die Menschen verstehen können und die ziemlich leicht in Maschinensprache übersetzt werden können.

  und

- **Syntax:** Dies ist eine Sprachstruktur (oder *Grammatik*), die es Menschen ermöglicht, diese C++-Befehle zu Programmen zu kombinieren, die tatsächlich etwas tun (nun ja, *vielleicht* etwas tun).

Sie können die Semantik auch als die Bausteine Ihrer C++-Programme auffassen und die Syntax als die korrekte Art und Weise, diese Bausteine zusammenzufügen.

Was ist ein Programm?

Ein C++-Programm ist eine Textdatei, die eine Folge von C++-Befehlen enthält, die den Regeln der C++-Grammatik entsprechend zusammengefügt worden sind. Diese Textdatei wird als *Quelldatei* bezeichnet (wahrscheinlich, weil sie die Quelle aller Frustrationen ist). Eine C++-Quelldatei hat die Namenerweiterung `.cpp`, so wie eine Microsoft-Word-Datei die Endung `.doc` oder eine MS-DOS-Batchdatei (wie Sie vielleicht noch wissen) die Endung `.bat` hat. Die Erweiterung `.cpp` ist nur eine Konvention.

Beim Programmieren in C++ geht es darum, eine Folge von Befehlen zu schreiben, die in ein Programm in Maschinensprache umgewandelt werden kann, das tatsächlich *tut*, was wir tun wollen. Das Ergebnis ist ein so genanntes *Executable*, eine von der Maschine *ausführbare* Datei, die standardmäßig die Erweiterung `.exe` hat. Das Verfahren, mit dem aus einem C++-Programm ein ausführbares Programm erstellt wird, wird als *Compilieren* oder *Building* be-

zeichnet (der feine Unterschied zwischen diesen beiden Konzepten wird in Kapitel 22 beschrieben).

Das hört sich einfach genug ein – wo also liegt das Problem? Nur weiter.

Wie programmiere ich?

Um ein Programm zu schreiben, benötigen Sie zwei spezielle Computerprogramme. Das eine Programm ist ein *Editor*, mit dem Sie Ihren Code schreiben, das heißt Ihre `.cpp`-Quelldatei erstellen. Das andere Programm ist ein Compiler, der Ihre Quelldatei in eine ausführbare `.exe`-Datei umwandelt, die die eigentliche Aufgabe erledigt (ein Arbeitsblatt einer Tabellenkalkulation öffnet, Musik spielt, eine Rakete steuert usw.).

Heutzutage fassen die Werkzeugentwickler normalerweise den Compiler und den Editor zu einem einzigen Paket zusammen, das als *Entwicklungsumgebung* bezeichnet wird. Nachdem Sie die Befehle Ihres Programms eingegeben haben, müssen Sie nur noch auf eine Schaltfläche klicken, um die ausführbare Datei zu erstellen.

Die verbreitetste C++-Umgebung ist ein Microsoft-Produkt: Visual C++ .NET (ausgesprochen »Visual-C-plus-plus-DOT-net«). Alle Programme in diesem Buch lassen sich mit Visual C++ .NET kompilieren und ausführen; doch viele Leser verfügen wahrscheinlich nicht über eine eigene Kopie von Visual C++ .NET – und bei Preisen ab etwa 130 Euro ist dies möglicherweise ein Problem.

Glücklicherweise *gibt es* public-domain C++-Umgebungen. Wir verwenden in diesem Buch eine dieser Umgebungen – die Dev-C++-Umgebung. Die beiliegende CD-ROM enthält eine neuere Version (Juni 2004) der Dev-C++-Umgebung. Die *allerneueste Version* finden Sie im Web unter `www.bloodshed.net`.

Sie können zahlreiche public-domain Programme aus dem Internet herunterladen. Jedoch sind einige dieser Programme nicht kostenlos; dann sollen (oder müssen) Sie eine (normalerweise kleine) Gebühr bezahlen, bevor Sie das Programm herunterladen (können). Um Dev-C++ nutzen zu können, *müssen Sie nicht* bezahlen, aber Sie können die Entwicklung unterstützen, wenn Sie wollen. Näheres dazu finden Sie auf der Website.

Ich habe die Programme in diesem Buch mit der Version 4.9.8.0 von Dev-C++ getestet; sie sollten auch mit anderen Versionen funktionieren. Auf meiner Website (`www.stephendavis.com`) finden Sie eine Liste der Probleme, die bei künftigen Versionen von Dev-C++ oder Windows auftreten könnten.

Dev-C++ ist keine fehlerverseuchte, eingeschränkte Ausgabe eines C++-Compilers einer obskuren Gruppe von Entwicklern, sondern eine ausgewachsene C++-Umgebung. Dev-C++ unterstützt die gesamte C++-Sprache und führt alle Programme aus diesem Buch (und aus jedem anderen C++-Buch) problemlos aus.

Dev-C++ erzeugt mit Windows kompatible 32-Bit-Programme, aber es ist nicht einfach, damit Programme zu erstellen, die das klassische Windows-Aussehen haben. Falls Sie dies tun wollen, müssen Sie schon das Geld für ein kommerzielles Paket wie Visual Studio .NET ausgeben. Nach diesen Vorbemerkungen lege ich Ihnen dringend ans Herz, erst die Beispiele in diesem Buch durchzuarbeiten, um C++ zu lernen, *bevor* Sie die Entwicklung von Windows-Programmen in Angriff nehmen. Es handelt sich um zwei verschiedene Dinge, die Sie (im Interesse Ihrer Geistesruhe) auch gedanklich auseinander halten sollten.

Führen Sie die Schritte im nächsten Abschnitt aus, um Dev-C++ zu installieren und Ihr erstes C++-Programm zu erstellen. Mit diesem Programm können Sie einen Temperaturwert eingeben und von Celsius-Graden in Fahrenheit-Grade umwandeln.

Die Programme in diesem Buch sind mit Visual C++ .NET und dem C++-Abschnitt von Visual Studio .NET (der im Wesentlichen dasselbe ist) kompatibel. In der Dokumentation von Visual C++ .NET finden Sie Anweisungen, um C++ zu installieren. Es stimmt zwar, dass Visual C++ .NET andere (und oft genauso schwer verständliche) Fehlermeldungen generiert, aber alles in allem wirkt das gesamte Territorium auf geheimnisvolle Weise vertraut. Auch wenn Sie ein anderes Gesangbuch verwenden, sollten Sie keine Schwierigkeiten haben, der Melodie zu folgen.

Dev-C++ installieren

Die beiliegende CD-ROM enthält die Version der Dev-C++-Umgebung, die zum Zeitpunkt der Übersetzung dieses Buches aktuell war.

Die Dev-C++-Umgebung wird als komprimierte, ausführbare Datei geliefert, die leicht zu installieren ist. Diese ausführbare Datei befindet sich im Verzeichnis `DevCPP` der beiliegenden CD-ROM. Installieren Sie Dev-C++ wie folgt:

1. **Navigieren Sie zu der Datei `devcpp4980.exe` und doppelklicken Sie auf die Datei. Alternativ können Sie (unter Windows) den Befehl Start|Ausführen auswählen.**

   - Wenn Sie auf die Datei doppelklicken, wird die Umgebung automatisch installiert. (Falls Sie Dev-C++ aus dem Web heruntergeladen haben, hat Ihre Version möglicherweise eine andere Versionsnummer als `4980`.)
   - Falls Sie den Befehl Start|Ausführen ausgewählt haben, geben Sie im Dialogfeld Ausführen den Befehl `x:\devcpp\devcpp4980` ein. Dabei repräsentiert *x* den Laufwerksbuchstaben Ihres CD-ROM-Laufwerks (normalerweise `D` oder `E` – falls der eine Buchstabe nicht funktioniert, versuchen Sie den anderen).

   Dev-C++ beginnt mit dem Hinweis (siehe Abbildung 1.1), dass Sie Dev-C++ nicht über eine möglicherweise vorhandene ältere Version installieren sollten. In diesem Fall sollten Sie die ältere Version zunächst deinstallieren, dann den Computer neu starten und die

Installation von vorne beginnen. (Eine Installation mit einer versteckten Drohung zu beginnen, ist zwar kein guter Anfang einer Beziehung, aber danach wird es freundlicher.)

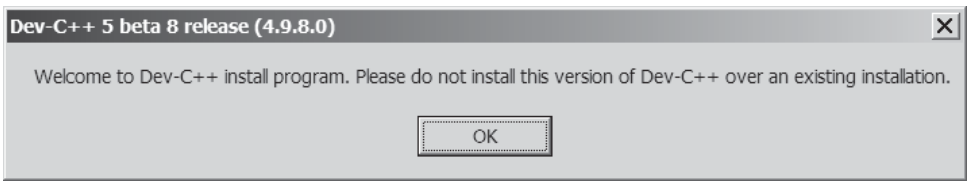


Abbildung 1.1: Sie müssen frühere Versionen von Dev-C++ deinstallieren, bevor Sie mit der Installation beginnen können.

2. **Falls Sie keine ältere Version von Dev-C++ deinstallieren müssen, können Sie mit Schritt 4 fortfahren; andernfalls klicken Sie auf OK und im folgenden Dialogfeld, das die Lizenzvereinbarung anzeigt, auf CANCEL, um die laufende Installation abzubrechen.**

Falls Sie vorher noch nie von Dev-C++ gehört haben, können Sie die Warnmeldung einfach ignorieren.

3. **Falls Sie bei diesem Schritt sind, deinstallieren Sie eine ältere Version: Öffnen Sie den DEV-CPP-Ordner und doppelklicken Sie auf die Datei `uninstall.exe` in diesem Ordner.**

   Das Deinstallationsprogramm wird ausgeführt und macht Platz für die neue Installation; die Lizenzvereinbarung (englisch *End User Legal Agreement*, kurz *EULA*) wird angezeigt.

4. **Lesen Sie die EULA und klicken Sie dann auf I AGREE, wenn Sie mit ihren Bedingungen leben können.**

   Wenn Sie CANCEL drücken, wird die Installation abgebrochen. Wenn Sie die Lizenzvereinbarung akzeptieren, wird ein Dialogfeld mit einigen Installationsoptionen geöffnet (siehe Abbildung 1.2). Bis auf die folgenden zwei Ausnahmen sind die Standardeinstellungen harmlos:

   - Die Option MINGW COMPILER SYSTEM... muss aktiviert sein.
   - Die Option ASSOCIATE C AND C++ FILES TO DEV-C++ bedeutet, dass bei einem Doppelklick auf eine `.cpp`-Datei automatisch Dev-C++ statt eines anderen Programms (wie beispielsweise Visual C++ .NET) geöffnet wird. Es ist möglich, aber schwierig, diese Verknüpfung aufzuheben.

Sie sollten diese Option nicht aktivieren, wenn Sie auch Visual Studio.NET installiert haben. Dev-C++ und Visual Studio.NET koexistieren friedfertig auf derselben Maschine, aber Sie sollten nicht in die Änderungen eingreifen, die Visual Studio vorgenommen hat. Sie können Ihre `.cpp`-Dateien immer noch mit Dev-C++ öffnen, indem Sie mit der rechten Maustaste auf die Datei klicken und dann ÖFFNEN MIT auswählen. Persönlich ziehe ich diese Option vor, selbst wenn Visual Studio.

NET installiert ist. Sie verursacht keine Probleme, und Dev-C++ startet *erheblich* schneller als Visual Studio.

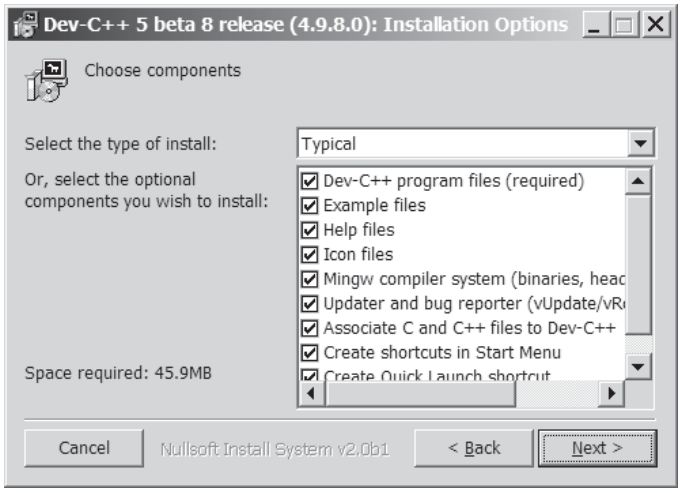


Abbildung 1.2: Die standardmäßigen Installationsoptionen sollten für die meisten Anwender akzeptabel sein.

5. Klicken Sie auf NEXT.

Das Installationsprogramm fragt Sie, wo Dev-C++ installiert werden soll (siehe das Dialogfeld in Abbildung 1.3).

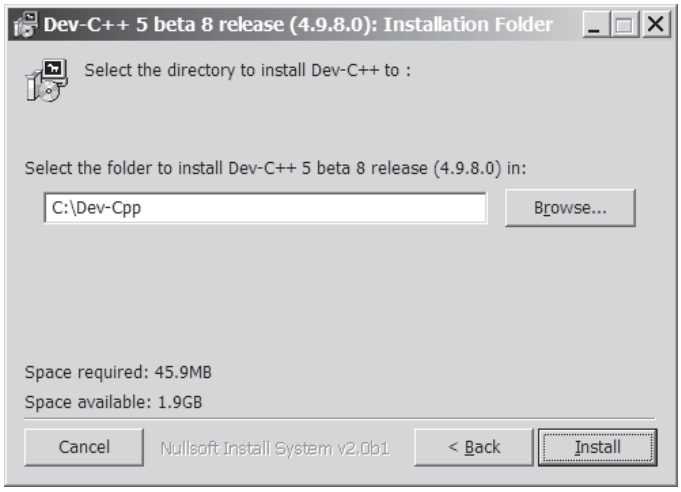


Abbildung 1.3: Den Speicherort für die Dev-C++-Umgebung festlegen

6. **Akzeptieren Sie das Standardverzeichnis C:\Dev-Cpp.**

Sie sollten Dev-C++ nicht in einem Verzeichnis installieren, dessen Name Leerzeichen enthält (wie beispielsweise das englische Programmverzeichnis \Program Files). Dev-C++ hat mit solchen Verzeichnissen Probleme. Ich bin dieser Sache nicht weiter nachgegangen, aber ich glaube, dass Sie jeden anderen Verzeichnisnamen ohne Sonderzeichen (außer »_«) verwenden können. Es ist sicherer, den Standardnamen zu verwenden.

7. **Vergewissern Sie sich, dass der Speicherplatz für das Programm ausreicht.**

   Die Dev-C++-Umgebung verwendet nur magere 45 MB, aber ein Profi prüft lieber noch einmal nach.

8. **Klicken Sie auf Install.**

   Zunächst scheint nichts zu passieren. Dann kopiert Dev-C++ zahlreiche Dateien in das Dev-Cpp-Verzeichnis. Es werden *absolut keine* Dateien in das Windows-Home-Verzeichnis eingefügt! Abbildung 1.4 zeigt das Endergebnis.

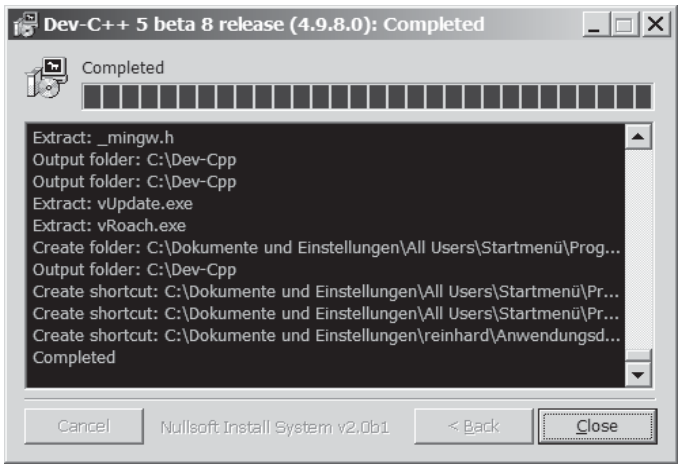


Abbildung 1.4: Bei der Dev-C++-Installation werden zahlreiche, meist kleine Dateien entpackt.

Während der Installation zeigt Dev-C++ ein Dialogfeld an, das fragt, ob Sie das Programm for all users (für alle Benutzer) installieren wollen, nachdem die Dateien auf Ihre Festplatte kopiert worden sind. Diese Frage bedeutet: Wollen Sie einer anderen Person, die sich bei Ihrem Computer anmeldet, erlauben, Dev-C++ auszuführen oder nicht? (In meinem Fall habe ich mit Yes geantwortet.)

9. **Wählen Sie, ob Dev-C++ für alle Benutzer installiert werden soll, und klicken Sie dann auf Close, um die Installation des Pakets abzuschließen.**

   Dev-C++ startet sofort, so dass Sie seine Optionen an Ihre Anforderungen anpassen können. (Jawohl, es bleibt noch etwas zu tun; aber das wussten Sie ja. Nur weiter.)

10. Dev-C++ zeigt Ihnen zunächst eine Meldung, die anzeigt, wo diverse Konfigurationsdateien gespeichert werden. Lesen Sie die Informationen und klicken Sie dann auf OK.

Das Dialogfeld DEV-C++ FIRST TIME CONFIGURATION wird geöffnet (siehe Abbildung 1.5).

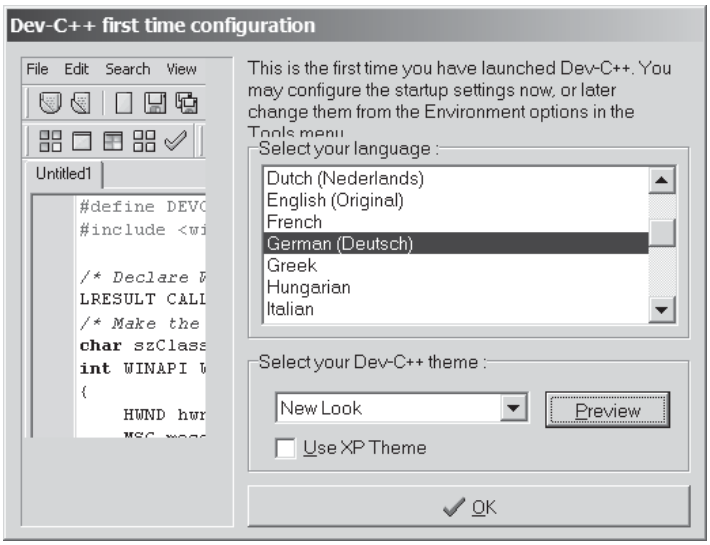


Abbildung 1.5: Die Auswahl der Sprache bei der Erstinstallation

11. Wählen Sie im Listenfeld SELECT YOUR LANGUAGE die Sprache GERMAN (DEUTSCH) aus, lassen Sie die anderen Einstellungen unverändert und klicken Sie auf OK.

Die Optionen einstellen

Wenn Sie nur ein wenig Zeit mit der Installation von Software verbracht haben, wissen Sie wahrscheinlich, dass die Einstellung von Optionen eine Sache für sich ist. In diesem Fall verfügt Dev-C++ über zwei Optionen, die Sie einstellen müssen, bevor Sie das Programm benutzen können:

1. **Wählen Sie WERKZEUGE|COMPILER OPTIONEN.**

   Sie können diese Einstellungen jederzeit ändern, warum also nicht gleich jetzt?

2. **Wählen Sie die Registerkarte EINSTELLUNGEN.**

3. **Wählen Sie im Menü auf der linken Seite den Eintrag CODE ERZEUGUNG.**

   Setzen Sie die Option ERLAUBE AUSNAHMEBEHANDLUNG auf YES (siehe Abbildung 1.6). (Mit dem kleinen Pfeil ganz rechts neben der Option können Sie die beiden möglichen Einstellungen auswählen.)

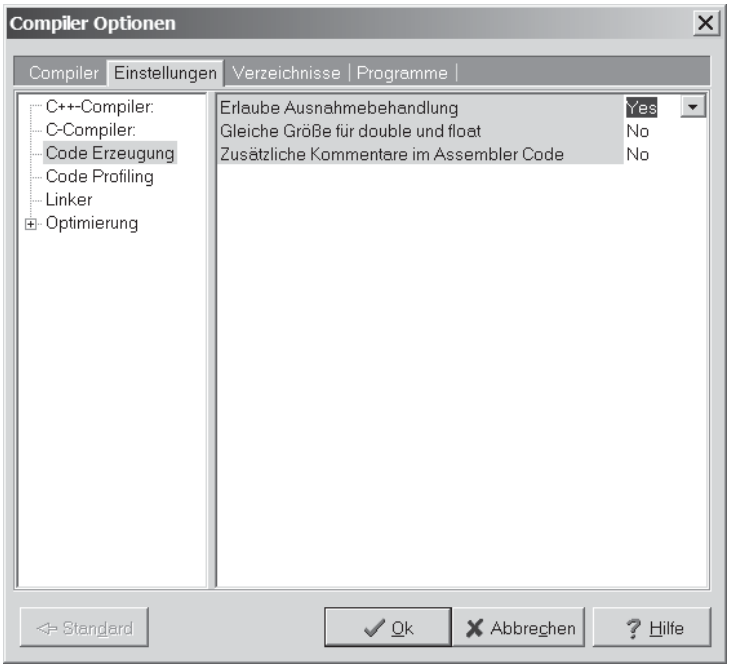


Abbildung 1.6: Die Option ERLAUBE AUSNAHMEBEHANDLUNG muss aktiviert sein.

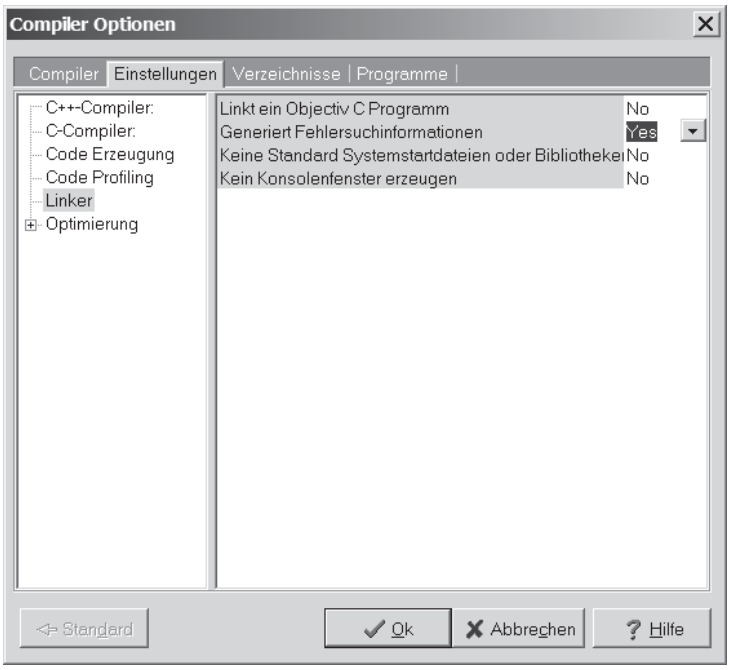


Abbildung 1.7: Die Option GENERIERT FEHLERSUCHINFORMATIONEN muss aktiviert sein.

4. **Wählen Sie im Menü auf der linken Seite den Eintrag Linker.**

   Setzen Sie die Option Generiert Fehlersuchinformationen auf Yes (siehe Abbildung 1.7). (Mit dem kleinen Pfeil ganz rechts neben der Option können Sie die beiden möglichen Einstellungen auswählen.)

5. **Klicken Sie auf OK.**

   Jetzt ist die Installation vollständig! (Ihre Optionen werden automatisch gespeichert.)

Ihr erstes C++-Programm erstellen

In diesem Abschnitt erstellen Sie Ihr erstes C++-Programm. Zunächst geben Sie den C++-Code in eine Datei namens `CONVERT.cpp` ein, dann wandeln Sie den C++-Code in ein ausführbares Programm um.

Den C++-Code eingeben

Der erste Schritt, um ein C++-Programm zu erstellen, besteht darin, C++-Anweisungen mit einem Texteditor einzugeben. Die Dev-C++-Benutzerschnittstelle besteht hauptsächlich aus einem Programmeditor, der speziell für das Schreiben von C++-Programmen entworfen wurde.

1. **Klicken Sie auf Start|Programme|Bloodshed Dev-C++|Dev-C++, um die Dev-C++-Entwicklungsumgebung zu starten.**

   Die Dev-C++-Benutzerschnittstelle sieht im Wesentlichen wie die jedes anderen Windows-Programms aus – vielleicht etwas uneleganter, aber das Aussehen einer Windows-Anwenung ist unverkennbar.

   Der Start erfordert viele Klicks – für meinen Geschmack zu viele. Ich ziehe es vor, eine Verknüpfung auf dem Desktop zu erstellen. Um eine Verknüpfung zu erstellen, doppelklicken Sie auf Arbeitsplatz, dann doppelklicken Sie auf Lokaler Datenträger (C:), und schließlich doppelklicken Sie auf Dev-Cpp – puh! Klicken Sie mit der rechten Maustaste auf die Datei `devcpp.exe` und wählen Sie in dem Kontextmenü den Befehl Verknüpfung erstellen. Ziehen Sie dann die Verknüpfung zu der Datei `devcpp.exe` auf Ihren Desktop (oder an eine andere leicht zugängliche Stelle). Ab jetzt können Sie Dev-C++ einfach mit einem Doppelklick auf diese Verknüpfung starten.

2. **Wählen Sie Datei|Neu|Quelldatei.**

   Dev-C++ öffnet ein leeres Fenster, in dem Sie Ihren neuen Code eingeben werden. Keine Bange, wenn Sie nicht wissen, was Sie dort eingeben sollen – deswegen haben Sie ja dieses Buch gekauft.

3. **Geben Sie das folgende Programm genau in der angegebenen Form ein.**

Sie brauchen sich jedoch nicht zu sehr um die Einrückung oder die Anzahl der Leerstellen zu kümmern – es spielt keine Rolle, ob eine Zeile um zwei oder drei Stellen eingerückt ist oder ob zwischen zwei Wörtern ein oder zwei Leerzeichen stehen. In C++ spielt jedoch die Groß- und Kleinschreibung eine Rolle, so dass Sie darauf achten müssen, dass alles in Kleinbuchstaben geschrieben ist.

Wenn Sie schummeln wollen, können Sie die Datei `Conversion.cpp` von der beiliegenden CD-ROM aus dem Verzeichnis `\Cpp_Programme\Kap01` kopieren.

```
//  Programm, um die Temperatur von Grad Celsius
//  in Grad Fahrenheit umzurechen:
//  Fahrenheit = Celsius  * (212 - 32)/100 + 32
//
#include <cstdio>
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int nNumberofArgs, char* pszArgs[])
{

  // die Temperatur in Celsius eingeben
  int celsius;
  cout << "Geben Sie die Temperatur in Grad Celsius ein:";
  cin >> celsius;

  // den Umrechnungsfaktor für Celsius
  // in Fahrenheit berechnen
  int factor;
  factor = 212 - 32;

  // mit dem Umrechnungsfaktor Grad Celsius
  // in Grad Fahrenheit umrechnen
  int fahrenheit;
  fahrenheit = factor * celsius/100 + 32;

  // das Ergebnis (und einen Zeilenumbruch) ausgeben
  cout << "entspricht Grad Fahrenheit:";
  cout << fahrenheit << endl;

  // warten, bis der Benutzer den Wert gelesen hat
  // und das Programm beendet
  system("PAUSE");
  return 0;
}
```

4. **Wählen Sie Datei|Speichern unter. Geben Sie den Programmnamen ein und drücken Sie auf ↵.**

   Ich weiß, dass dies vielleicht nicht besonders aufregend aussieht, aber Sie haben gerade Ihr erstes C++-Programm erstellt!

Für dieses Buch habe ich einen Ordner namens `Cpp_Programme` und darin einen Unterordner namens `Kap01` erstellt. Schließlich habe ich das Programm unter dem Namen `Conversion.cpp` gespeichert. Beachten Sie, dass Dev-C++ bei Verzeichnisnamen, die Leerzeichen enthalten, nicht korrekt funktioniert. (Glücklicherweise gibt es keine Probleme mit Namen, die mehr als acht Zeichen lang sind!)

Ihr Programm erstellen

Nachdem Sie Ihre C++-Quelldatei `Conversion.cpp` auf die Festplatte gespeichert haben, können Sie daraus den ausführbaren Maschinencode erstellen.

Wählen Sie zu diesem Zweck den Menübefehl Ausführen|Kompilieren oder drücken Sie auf F9. Alternativ können Sie sogar auf das hübsche kleine Symbol mit den vier farbigen Quadraten in der Symbolleiste klicken (verwenden Sie die QuickInfo oder ToolTips, um das betreffende Symbol zu identifizieren). Als Reaktion darauf öffnet Dev-C++ das Dialogfeld Compile Progress. Zunächst passiert nichts (psst ... der Computer denkt). Nach einigen Sekunden scheint Dev-C++ loszulegen und Ihr Programm mit Schwung zu kompilieren. Wenn alles gut geht, wird das Ergebnis in dem Dialogfeld angezeigt (siehe Abbildung 1.8).

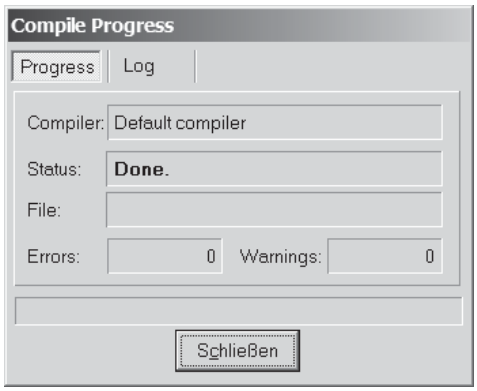


Abbildung 1.8: Der Benutzer wird mit einer einfachen Done-Meldung belohnt, wenn das Programm fehlerfrei ist.

Dev-C++ erzeugt eine Meldung, wenn es in Ihrem C++-Programm einen Fehler findet – und Fehler beim Programmieren kommen etwa so häufig vor wie Sand am Meer. Wahrscheinlich werden Sie selbst, wenn Sie ein so einfaches Programm wie `Conversion.cpp` eingeben, auf zahlreiche Warnungen und Fehlermeldungen stoßen. Um die Art der Fehleranzeige zu demonstrieren, wollen wir Zeile 17 von `cin >> celsius;` in `cin >>> celsius;` ändern.

Dieser Verstoß scheint unschuldig genug zu sein – und in Ihren und vielleicht in meinen Augen sogar verzeihlich, aber nicht für C++. Dev-C++ öffnet eine Registerkarte namens Compiler (siehe Abbildung 1.9). Die Meldung parse error before '> ist vielleicht etwas knapp, aber aussagekräftig. Um die Meldung loszuwerden, entfernen Sie das überflüssige > und kompilieren Sie das Programm neu.

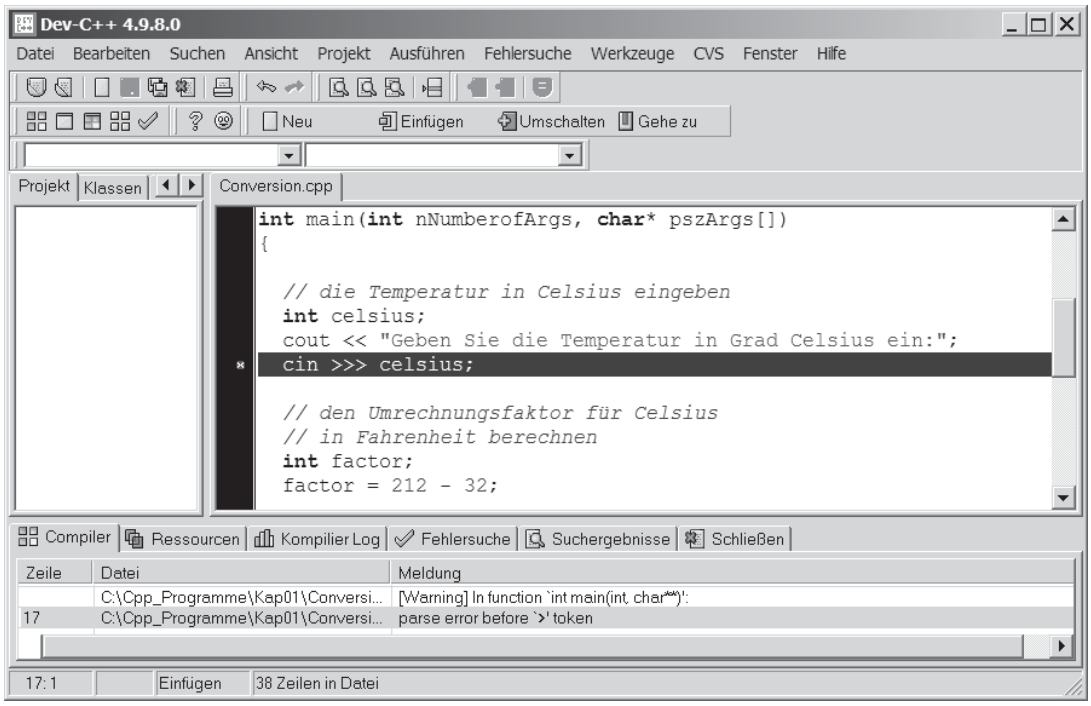


Abbildung 1.9: Böse kleine Programme erzeugen in dem Compiler-Fenster Fehlermeldungen.

Der Terminus *parse* bedeutet die Umwandlung der C++-Befehle in eine Form, die der Teil des Compilers, der den Maschinencode generiert, weiterverarbeiten kann.

Es gab einmal eine Sprache, die versuchte, einfache Fehler dieser Art automatisch für Sie zu beheben. Aus eigener Erfahrung kann ich Ihnen sagen, dass dies eine Zeitverschwendung war – weil der Compiler bis auf wenige sehr einfache Fälle fast immer daneben lag. Doch wenigstens wies er mich auf das Problem hin, so dass ich es selbst beheben konnte.

Warum ist C++ so penibel?

In dem genannten Beispiel könnte C++ sofort – und ohne Zweifel – sagen, dass ich einen Fehler gemacht habe. Doch wenn C++ herausfinden kann, was ich falsch gemacht habe, warum behebt es nicht einfach das Problem und macht weiter?

Die Antwort ist einfach, hat aber weitreichende Konsequenzen. C++ *denkt*, dass ich das Symbol >> falsch eingetippt habe, aber dies könnte nicht richtig sein. Was wie ein falsch eingetippter Befehl aussieht, könnte tatsächlich auch ein anderer, vollkommen unzusammenhängender Fehler sein. Hätte der Compiler einfach das Problem korrigiert, hätte C++ möglicherweise das eigentliche Problem verdeckt.

Einen Fehler zu finden, der in einem Programm vergraben ist, das ohne Fehlermeldungen kompiliert wird, ist schwierig und zeitaufwändig. Es ist auf jeden Fall viel besser, es dem Compiler zu überlassen, den Fehler zu finden, falls dies überhaupt möglich ist. Einen Compiler-Fehler zu generieren, verschwendet die Zeit des Computers – doch mich zu zwingen, einen Fehler zu finden, den C++ hätte abfangen können, verschwendet *meine* Zeit. Was meinen Sie wohl, was mir lieber ist?

Ihr Programm ausführen

Jetzt können wir Ihre neue Kreation ausführen ... das heißt, Ihr Programm laufen lassen. Sie werden die Programmdatei `CONVERT.EXE` ausführen und einige Eingaben machen, um zu sehen, wie gut es funktioniert.

Um das Umrechnungsprogramm auszuführen, klicken Sie auf Ausführen|Ausführen, oder drücken Sie auf Strg+F10. (Ich habe keine Ahnung, wie die Funktionstasten ausgewählt wurden. Meiner Meinung nach würde eine so häufige Aktion wie das Ausführen eines Programms eine eigene Funktionstaste rechtfertigen – ohne Strg oder ⇧ festhalten zu müssen, aber vielleicht bin ich ja zu anspruchsvoll.)

Sofort öffnet sich ein Fenster, und Sie werden aufgefordert, eine Temperatur in Grad Celsius einzugeben. Geben Sie eine bekannte Temperatur wie beispielsweise 100 Grad ein. Nachdem Sie auf ↵ gedrückt haben, zeigt das Programm die entsprechenden Grade Fahrenheit (212) an (siehe Abbildung 1.10):

Die Meldung Drücken Sie eine beliebige Taste... gibt Ihnen Gelegenheit, die Werte in Ruhe zu studieren, bevor das Programm beendet wird. Wenn Sie auf ↵ drücken, wird das Fenster geschlossen, und sein Inhalt verschwindet. Glückwunsch! Sie haben gerade Ihr erstes C++-Programm eingegeben, erstellt und ausgeführt.

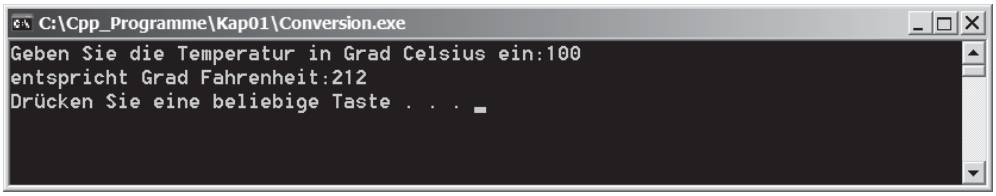


Abbildung 1.10: Eine Umwandlung von Grad Celsius in Grad Fahrenheit

Dev-C++ ist nicht Windows

Beachten Sie, dass Dev-C++ nicht wirklich dafür bestimmt ist, Windows-Programme zu entwickeln. Theoretisch können Sie eine Windows-Anwendung mit Dev-C++ schreiben, aber es ist nicht einfach. (Mit Visual Studio .NET geht dies *sehr viel* einfacher.)

Windows-Programme zeigen sich dem Anwender in einer sehr stark optisch betonten Form, die auf Bildschirmfenstern basiert. CONVERT.EXE ist ein 32-Bit-Programm, das *unter* Windows ausgeführt wird, aber es ist kein »Windows«-Programm in Sinne einer typischen, visuell betonten Windows-Anwendung.

Es macht nichts, wenn Sie nicht wissen, was *32-Bit-Programm* bedeutet. Wie ich bereits weiter oben erwähnt habe, ist das Schreiben von Windows-Programmen nicht das Thema dieses Buches. Die C++-Programme, die Sie in diesem Buch schreiben, haben eine so genannte *Kommandozeilen-* oder *Befehlszeilen-Schnittstelle* (engl. *command line interface*), die in einem MS-DOS-Fenster ausgeführt wird.

Angehende Windows-Programmierer sollten nicht verzweifeln – sie haben ihr Geld nicht verschwendet. C++ zu lernen, ist eine Voraussetzung dafür, Windows-Programme zu schreiben. Ich bin der Meinung, dass beide Themen separat gemeistert werden sollten, und zwar zuerst C++ und dann Windows.

Dev-C++-Hilfe

Dev-C++ verfügt über ein Hilfemenü. Um das typische Hilfe-Dialogfeld von Windows zu öffnen, wählen Sie den Menübefehl HILFE|DEV C++ HILFE. Die Hilfe umfasst verschiedene Aspekte des Dev-C++-Entwicklungspaketes, aber nicht viel mehr. Insbesondere fehlen Informationen über die C++-Sprache selbst. Durch einen Klick auf ein Thema können Sie die einschlägigen Informationen abrufen.

Nähere Anmerkungen zu Ihrem Programm

Daten in das Programm eines anderen Programmierers einzugeben, ist damit zu vergleichen, einem anderen Autofahrer beim Autofahren zuzuschauen. Der eigentliche Spaß beginnt erst, wenn Sie sich selbst hinters Steuer setzen. Bei Programmen ist es ähnlich wie bei Autos: Alle

Autos verhalten sich bis auf kleinere Unterschiede gleich – wenigstens wenn wir Autos aus Frankreich außen vor lassen; aber grundsätzlich ist diese Aussage schon richtig. Autos haben im Wesentliche dieselbe Grundstruktur – vor Ihnen gibt es ein Steuerrad, unter Ihnen ein Sitz, über Ihnen ein Dach usw.

Analog dazu sind auch alle C++-Programme nach einem gemeinsamen Muster aufgebaut. Dieses Muster zeigt sich bereits in Ihrem allerersten Programm. Wir wollen uns das Umwandlungsprogramm noch einmal im Hinblick auf die Elemente anschauen, die allen Programmen gemeinsam sind.

Das gemeinsame Gerüst aller C++-Programme

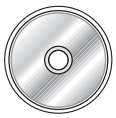
Jedes C++-Programm, das Sie für dieses Buch schreiben, hat dieselbe folgende Grundstruktur:

```
//  Template - stellt eine Schablone zur Verfügung, die als
//             Ausgangspunkt verwendet wird
//
// Die folgenden Include-Dateien definieren die Mehrzahl der
// Funktionen, die von allen Programmen benötigt werden.
#include <cstdio>
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int nNumberofArgs, char* pszArgs[])
{
    // Hier beginnt Ihr C++-Code.

    // Warten, bis der Benutzer bereit ist, das Programm zu beenden,
    // damit er die Ergebnisse des Programms studieren kann.
    system("PAUSE");
    return 0;
}
```

Wir wollen nicht auf die vielen langweiligen Einzelheiten eingehen. Nur so viel: Die Ausführung beginnt mit dem Code, der unmittelbar nach der öffnenden geschweiften Klammer nach `main()` beginnt und vor der schließenden geschweiften Klammer von `main()` endet.



Der Code dieser Schablone (Template) befindet sich auf der beiliegenden CD-ROM im Hauptordner `Cpp_Programme` in einer Datei namens `Template.cpp`.

Sourcecode durch Kommentare erläutern

Die ersten wenigen Zeilen in `Conversion.cpp` scheinen aus Fließtext ohne eine spezielle Form zu bestehen. Entweder ist dieser Code für menschliche Augen bestimmt, oder C++ ist sehr viel intelligenter, als mir bewusst ist. Diese ersten sechs Zeilen werden als *Kommentare* bezeichnet. Mit Hilfe von *Kommentaren* kann der Programmierer erklären, was ein bestimmter Codeabschnitt tun soll. Der Compiler ignoriert Kommentare. Programmierer (das heißt *gute* Programmierer) achten auf sie.

Ein C++-Kommentar beginnt mit einem doppelten Schrägstrich (`//`) und endet mit einem Zeilenumbruch. Kommentare dürfen beliebige Zeichen enthalten. Ein Kommentar kann so lang sein, wie Sie wollen, aber üblicherweise wird die Länge auf 80 Zeichen beschränkt. Früher waren Bildschirme standardmäßig 80 Zeichen breit. Einige Drucker drucken standardmäßig immer noch Zeilen, die 80 Zeichen lang sind. Auch heutzutage ist durchaus noch sinnvoll, keine Zeile länger als 80 Zeichen zu machen (sie sind leichter zu lesen, strengen die Augen weniger an usw.).

Ein Zeilenumbruch wurde früher im Zeitalter der Schreibmaschine auch als *Wagenrücklauf* bezeichnet; damals wurde die Eingabe von Zeichen in eine Maschine auch als *Eintippen* statt als *Tastatureingabe* bezeichnet. Ein *Zeilenumbruch* ist das Zeichen, das eine Befehlszeile beendet.

In C++ gibt es auch eine zweite Form von Kommentaren, die durch die Zeichen `/*` eingeleitet und die Zeichen `*/` beendet werden; alles, was zwischen diesen Begrenzern steht, wird ignoriert. Jedoch wird diese Form von Kommentaren in C++ normalerweise nicht mehr verwendet. (Später in diesem Buch beschreibe ich einen Fall, in dem diese Art des Kommentars verwendet wird.)

Es mag seltsam erscheinen, dass es in C++ (oder in anderen Programmiersprachen) einen Befehl gibt, der von dem Computer ignoriert wird. Jedoch gibt es in allen Computersprachen eine Art von Kommentar. Es ist wichtig, dass der Programmierer erklären kann, was er beabsichtigte, als er den Code schrieb; denn möglicherweise sind seine Absichten für einen Kollegen, der das Programm aufgreift und versucht, es zu verstehen, nicht so leicht durchschaubar. Tatsächlich könnte auch der Programmierer selbst vergessen haben, was er mit dem Code bewirken wollte, wenn er sich sein Programm Monate später noch einmal anschaut, nachdem er den Code geschrieben hat, und keinen Hinweis hinterlassen hat.

Programme bestehen aus C++-Befehlen

Alle C++-Programme bestehen aus so genannten C++-*Anweisungen*. In diesem Abschnitt erhalten Sie einen Überblick über die Anweisungen, aus denen das Programmgerüst besteht, das von dem Programm `Conversion.cpp` verwendet wird.

Eine *Anweisung* ist eine einzelner Satz von Befehlen. Alle Befehle außer Kommentaren enden mit einem Semikolon. (Es gibt einen Grund dafür, warum Kommentare nicht mit einem Semikolon abgeschlossen werden, aber er ist dunkel. Meiner Meinung nach *sollten* auch Kommen-

tare aus Gründen der Konsistenz mit einem Semikolon abschlossen werden. Warum mich niemand gefragt hat, ist mir schleierhaft.)

Die Ausführung eines Programms beginnt mit dem ersten C++-Befehl nach der öffnenden geschweiften Klammer und wird Befehl für Befehl bis zum Ende des Listings fortgesetzt.

Wenn Sie das Programm studieren, werden Sie feststellen, dass im ganzen Programm Leerzeichen, Tabulatorzeichen und Zeilenumbrüche vorkommen. Tatsächlich habe ich in diesem Programm nach jedem Befehl einen Zeilenumbruch eingefügt. Diese Zeichen werden zusammenfassend als *Whitespace* (»weißer Raum«) bezeichnet, weil sie auf dem Bildschirm nicht angezeigt werden.

Sie können Whitespace überall in ein Programm einfügen, um seine Lesbarkeit zu verbessern – außer in der Mitte eines Wortes:

```
Sehen Sie, wa

s ich meine?
```

Im Gegensatz zu Whitespace spielt die Groß- und Kleinschreibung in C++ eine Rolle. Tatsächlich nimmt C++ die Schreibweise sehr genau. Die Variablen `fullspeed` und `FullSpeed` haben nichts miteinander zu tun. Während die Anweisung `int` in C++ genau definiert ist, kann C++ mit `INT` nichts anfangen.

Deklarationen schreiben

Die Zeile `int nCelsius;` ist eine Deklarationsanweisung. Eine *Deklaration* ist ein Befehl, der ein Variable definiert. Eine *Variable* ist ein »Behälter« für einen Wert eines bestimmten Typs. Eine Variable enthält einen *Wert* wie beispielsweise eine Zahl oder ein Zeichen.

Der Terminus *Variable* ist aus algebraischen Formeln der folgenden Art abgeleitet:

```
x = 10
y = 3 * x
```

Im zweiten Ausdruck wird `y` gleich `3` mal `x` gesetzt; aber was ist `x`? Die Variable `x` hat die Funktion eines Platzhalters für einen Wert. In diesem Fall hat `x` den Wert `10`, aber wir hätten den Wert von `x` auch auf `20` oder `30` oder `-1` setzen können. Die zweite Formel ist unabhängig vom Wert von `x` gültig.

In der Algebra dürfen Sie mit einer Anweisung wie beispielsweise `x = 10` anfangen. In C++ muss der Programmierer zuerst die Variable `x` definieren, bevor er sie benutzen kann.

In C++ hat eine Variable einen Typ und einen Namen. Die Variable, die in Zeile 11 definiert wird, heißt `celsius`; die Deklaration legt fest, dass es sich um eine Ganzzahl handelt. (Warum die Erfinder der Sprache nicht einfach `integer` statt `int` gewählt haben, ist mir unverständlich. Es gehört einfach zu den Dingen, mit denen zu leben Sie lernen müssen.)

Der Name einer Variablen hat für C++ keine besondere Bedeutung. Eine Variable muss mit einem der Buchstaben von `A` bis `Z` oder `a` bis `z` beginnen. Alle folgenden Zeichen müssen aus einem Buchstaben, einer Ziffer (`0` bis `9`) oder einem Unterstreichungszeichen (_) bestehen. Variablennamen können so lang sein, wie Sie wollen.

Per Konvention beginnen Variablennamen mit einem Kleinbuchstaben. Jedes neue Wort *innerhalb* einer Variablen beginnt mit einem Großbuchstaben – beispielsweise `meineVariable`.

Sie sollten versuchen, kurze, aber aussagekräftige Variablennamen zu formulieren. Sie sollten Namen wie `x` vermeiden, weil `x` keine spezielle Bedeutung hat. Ein Variablenname wie `laengeDerLinie` ist viel aussagekräftiger.

Ausgaben generieren

Die Zeilen, die mit `cout` und `cin` beginnen, werden als *Input/Output-Befehle* (Eingabe-/Ausgabebefehle, oft abgekürzt als I/O-Befehle) bezeichnet. (Programmierer lieben wie alle Ingenieure Abkürzungen.)

Der erste I/O-Befehl enthält die Anweisung, die Phrase `Geben Sie die Temperatur in Grad Celsius ein:` auf `cout` (ausgesprochen »ßi-aut«) auszugeben. `cout` ist der Name des Standardausgabegerätes von C++. In diesem Fall ist Ihr Monitor das Standardausgabegerät von C++.

Die nächste Zeile bewirkt genau das Gegenteil. Sie besagt: *Lies einen Wert von dem C++-Eingabegerät ein und speichere ihn in der ganzzahligen Variablen* `celsius`. Das C++-Eingabegerät ist normalerweise die Tastatur. Hier handelt es sich um das C++-Gegenstück zu der algebraischen Formel $x = 10$, die ich gerade erwähnt habe. Im Rest des Programmes hat `celsius` den Wert, den der Benutzer hier eingibt.

Ausdrücke berechnen

Bis auf die einfachsten Programme führen alle Programme die eine oder andere Berechnung durch. In C++ ist ein *Ausdruck* ein Befehl, der eine Berechnung durchführt. Anders ausgedrückt: Ein Ausdruck ist ein Befehl, der *einen Wert hat*. Ein *Operator* ist eine Anweisung, die einen Wert generiert.

Beispielsweise gibt es in unserem Umrechnungsprogramm zwei Zeilen, in denen der Umrechnungsfaktor mit Hilfe der Variablen `factor` deklariert und dann berechnet wird. Diese spezielle Anweisung berechnet die Differenz von 212 und 32; der Operator ist das Minuszeichen (`-`), und der Ausdruck ist `212-32`.

Das Ergebnis eines Ausdrucks speichern

Die Umgangssprache kann sehr vieldeutig sein. Dies gilt auch für den Ausdruck *ist gleich*. Das Wort *gleich* kann bedeuten, dass zwei Dinge denselben Wert haben; beispielsweise waren zu DM-Zeiten 10 Pfennige gleich einem Groschen. Das Wort *gleich* kann aber auch eine Zuweisung bedeuten, wie beispielsweise in der mathematischen Formel *y gleich 3 mal x*.

Um Mehrdeutigkeiten zu vermeiden, sprechen C++-Programmierer von dem *Zuweisungsoperator,* der besagt: »*Speichere das Ergebnis des Ausdrucks auf der rechten Seite des Gleichheitszeichens in der Variablen auf der linken Seite.*« Programmierer sagen: »Der Variablen `factor` wird *der Wert* von 212 minus 32 *zugewiesen*.«

Sagen Sie nie: »Die Variable `factor` ist *gleich* 212 minus 32.« Nachlässige Programmierer verwenden manchmal diese Ausdrucksweise; aber Sie und ich wissen es besser.

Den Rest von Conversion.cpp untersuchen

Der zweite Ausdruck in `Conversion.cpp` ist etwas komplizierter. Er verwendet dieselben Operatoren, die Sie auch aus der Mathematik kennen: `*` für die Multiplikation, `/` für die Division und `+` für die Addition. In diesem Fall erfolgt die Berechnung jedoch mit Variablen und nicht mit einfachen Konstanten.

Der Wert, der in der Variablen `factor` enthalten ist (die unmittelbar zuvor berechnet wurde), wird mit dem Wert multipliziert, der in der Variablen `celsius` enthalten ist (und der über die Tastatur eingegeben wurde). Das Ergebnis wird durch 100 geteilt; dann wird 32 addiert. Das Ergebnis des gesamten Ausdrucks wird der ganzzahligen Variablen `fahrenheit` zugewiesen.

Die letzten beiden Befehle geben den String `entspricht Grad Fahrenheit:` sowie den Wert der Variablen `fahrenheit` auf dem Bildschirm aus – und zwar so schnell, dass der Benutzer kaum mitkriegt, was passiert.