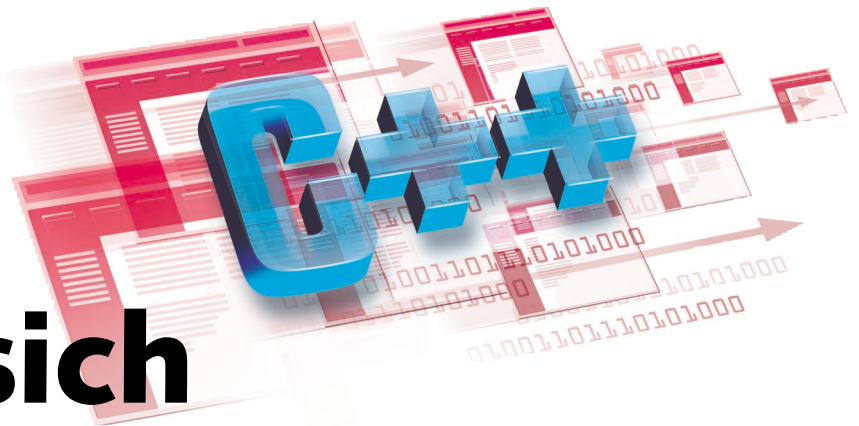




Hello World

VC++ meldet sich



Die **Entwicklungsumgebung** ist installiert – jetzt geht's ans **Eingemachte**, denn das erste **Programm** will geschrieben werden.

THOMAS WÖLFER

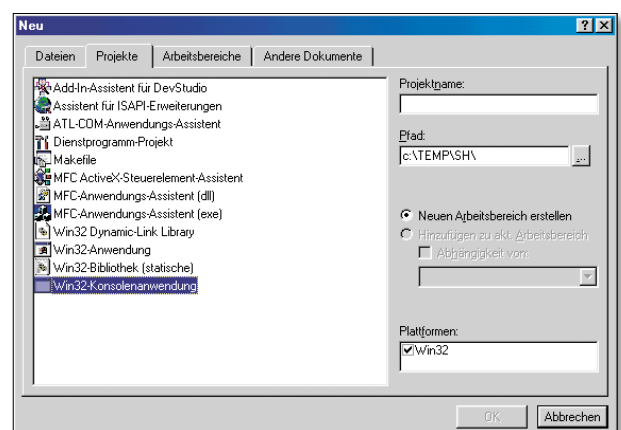
Das klassischerweise immer als Erstes programmierte Programm trägt den schönen Namen "Hello World" (Hallo Welt) – und zwar deshalb, weil dieses Programm genau diesen Text auf der Konsole anzeigt. Dieses Programm wird nun im Folgenden geschrieben oder – um genau zu sein – vom zugehörigen Wizard mehr oder minder automatisch erzeugt werden. Dabei erfahren Sie eine ganze Menge über die beteiligten Dateien und verwenden auch zum ersten Mal den VC++-Debugger.

Zunächst müssen Sie VC++ starten. Das Programm wird geladen und meldet sich mit einem "Tipp des Tages". Den können Sie entweder lesen oder ignorieren – auf jeden Fall haben Sie danach ein Fenster mit der IDE vor sich, das im Wesentlichen aus einem Fenster mit Menüleiste, Werkzeugleiste und Statusleiste besteht: Ansonsten ist das Fenster leer. Bei fol-

genden Sitzungen wird das allerdings anders aussehen, denn VC++ lädt per Default beim Start immer das letzte Projekt automatisch nach. In diesem Fall passiert noch nichts, denn Sie haben die IDE gerade erst installiert. Es gibt also kein "letztes" Projekt und damit auch nichts, was VC++ automatisch laden könnte.

■ Das erste Projekt

Zunächst muss ein Projekt angelegt werden. Dazu gehen Sie ins Datei-Menü und wählen dort den Befehl *Neu*. Damit können alle möglichen "neuen" Dinge erzeugt werden und unter anderem auch Projekte. Dazu erscheint der Dialog *Neu* der einige Reiter hat – Sie befinden sich aber automatisch auf dem richtigen

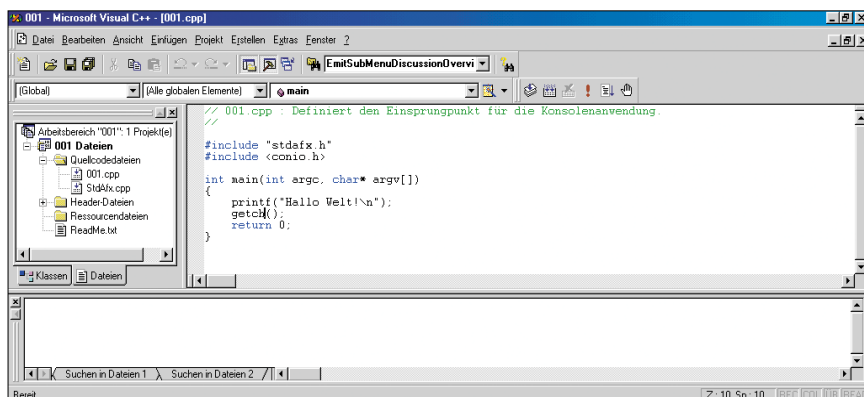


DAS ERSTE PROJEKT, das Sie anlegen, ist eines für eine Win32-Konsolenanwendung.

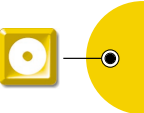
Reiter, nämlich dem mit dem Namen *Projekt*. Dort werden eine ganze Reihe von Projekt-Typen vorgeschlagen. Welche das im Einzelnen sind, finden Sie im Kasten "VC++ - die Projekt-Typen" erläutert.

Das Projekt, das Sie nun erzeugen wollen, ist eine *Win32 Konsolenanwendung*. Nachdem Sie diesen Typ ausgewählt haben, müssen Sie rechts oben im Dialog noch einen Namen und einen Pfad für das neue Projekt angeben. (Das Beispielprojekt auf der Heft-CD trägt den Namen 001.) Sie können natürlich irgendeinen Namen angeben. Nachdem aber der Name des Projekts für verschiedene Dinge zuständig ist, dient dieser Name zum Beispiel auch als Basis für verschiedene Dateien, die VC++ anlegt. Es ist deshalb im Rahmen der Nachvollziehbarkeit angebracht, wenn Sie als Namen ebenfalls 001 angeben. Bei Ihrer späteren Arbeit mit VC++ wird es dabei sicherlich besser sein, sinnvolle und beschreibende Namen zu wählen.

Wenn Sie dann OK klicken erscheint ein Dialog, auf dem Sie die Art Ihres



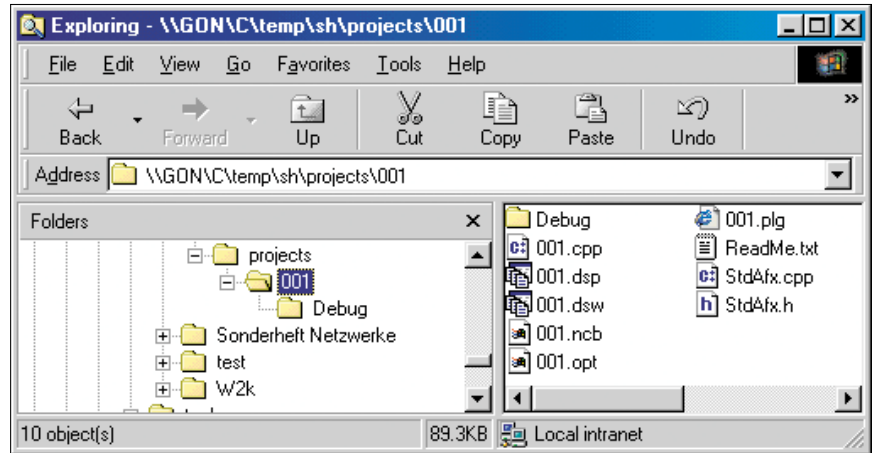
VISUAL C++ mit Ihrem ersten eigenen Quellcode: noch nicht sehr beeindruckend, aber voll funktionstüchtig.



Konsolen-Projektes auswählen können. Die verschiedenen Arten unterscheiden sich in diesem Fall im Wesentlichen durch den Code, den VC++ gleich erzeugen wird – allerdings sind die Unterschiede nicht sehr groß. Wählen Sie als Typ die *Hallo Welt Anwendung* aus, und klicken Sie dann auf *Fertigstellen*. VC++ zeigt nun nochmals an, welche Einstellungen gewählt wurden. Ein weiterer Klick auf *OK* bringt die IDE dazu, eine ganze Menge Funktionen in Gang zu setzen.

Was automatisch geschieht

Zunächst wird ein neues Projekt angelegt. Alle benötigten Compiler- und Linkeroptionen für eine Win32-Kommandozeilenanwendung sind darin bereits richtig gesetzt – und zwar sowohl für die Debug- als auch für die Release-Version. Die Debug-Version wird dabei aktiviert. Ferner legt VC++



BEIM ANLEGEN EINES NEUEN PROJEKTS und beim Übersetzen des Projekts erzeugt VC++ eine ganze Menge Dateien.

einige Dateien und allen voran eine Datei mit dem Namen 001.cpp an. Diese Datei enthält den eigentlichen Quellcode für das Projekt. Insgesamt werden die folgenden Dateien erzeugt:

- 001.cpp
- 001.dsp

- 001.wsp
- stdafx.h
- stdafx.cpp
- readme.txt

Die Datei 001.cpp enthält wie gesagt den eigentlichen Quellcode für das Programm, in dieser Datei werden Sie noch

VC++ – DIE PROJEKTTYPEN

Beim Anlegen eines neuen Projekts bietet VC++ eine Auswahl unter mehreren Projekttypen an. Hier sehen Sie eine Kurzübersicht, was diese Projekte im Einzelnen ausmachen:

Add-In-Assistent für DevStudio

Mit diesem Projekttyp erzeugen Sie Erweiterungen für das Developer-Studio, also für die VC++-IDE. Die IDE ist nicht nur mit Makros erweiterbar – Sie können VC++ auch mit nativ übersetztem Code um Funktionen erweitern. Dieser Projekttyp kümmert sich darum, dass alle dafür benötigten Compiler- und Linkerschalter richtig gesetzt sind.

Assistent für ISAPI-Erweiterungen

Dieser Assistent bereitet ein Projekt vor, mit dem Sie eine ISAPI-Erweiterung programmieren können. ISAPI steht dabei für "Internet Server API" (API=Application Program Interface). Mit anderen Worten: Mit diesem Projekt schreiben Sie Erweiterungen für den Internet Information Server.

ATL-COM-Anwendungs-Assistent

Mit diesem Assistenten erzeugen Sie ein Projekt, dessen Ziel die Schaffung eines neuen COM-Objekts ist. COM-Objekte sind Programme, die in Binärforn vorliegen und von anderen Programmen als Objekt benutzbar sind. Wenn Sie schon einmal mit VBS oder VB gearbeitet haben, kennen Sie COM-Objekte schon – das sind die Objekte, die Sie mit der CreateObject()-Funktion erzeugen können.

Makefile

Mit diesem Projekt wird ein einfaches Makefile angelegt. Makefiles steuern die

Art und Weise, wie ein Programm übersetzt und gelinkt wird – solange Sie immer nur innerhalb der VC++-IDE arbeiten, werden Sie keine Makefiles benötigen, denn diese sind speziell für die Verwendung auf der Kommandozeile gedacht.

MFC-ActiveX-Steuerelemente-Assistent

Mit dieser Projektart erzeugen Sie ein ActiveX Control mit Hilfe der MFC-Klassenbibliothek. ActiveX Controls kommen unter anderem aber nicht ausschließlich auf Webseiten zum Einsatz: Wenn Sie ein Programm schreiben möchten, das im Browser ausgeführt werden kann, sind ActiveX Controls für Sie unter Umständen das Richtige.

MFC-Anwendungs-Assistent (dll)

Mit diesem Projekttyp erzeugen Sie eine Anwendung für Windows, die die MFC-Klassenbibliothek verwendet aber in Form einer DLL vorliegt. Mit anderen Worten – ein solches Projekt erzeugt im Wesentlichen extern lagerbare Programmfunktionalität mit Hilfe von MFC.

MFC-Anwendungs-Assistent (exe)

Auch hier wird eine Anwendung erzeugt, und auch hier kommt die MFC-Klassenbibliothek zum Zuge. Allerdings ist das Resultat des Projekts keine Programmerweiterung, sondern ein eigenständiges Programm. Wenn Sie hauptsächlich normale Windows-Anwendungen programmieren möchten, ist dieser Projekttyp der, den Sie verwenden sollten.

Win32 Dynamic Link Library

Dieser Projekttyp erzeugt eine DLL, also eine Anwendungserweiterung – allerdings eine, die ohne die Hilfe von MFC auskommen muss. Stattdessen wird die ganz normale Win32-API ohne weitere Hilfsmittel verwendet.

Win32-Anwendung

Mit diesem Projekttyp erzeugen Sie ein eigenständiges Windows-Programm, allerdings ohne die Hilfe der MFC. Wenn irgend möglich sollten Sie das besser bleiben lassen, denn MFC ist für die meisten Fälle der sinnvollere und einfachere Weg, ein Windows-Programm zu schreiben.

Win32-Bibliothek (statisch)

Programmcode kann nicht nur in .EXE- und .DLL-Dateien abgelegt werden, sondern auch in statischen Bibliotheken vorliegen. Diese Bibliotheken können dann vom Linker mit dem übrigen Programmcode zusammengebunden werden. Wenn Sie zum Beispiel einen Satz an Funktionen haben, den Sie immer wieder in verschiedenen Programmen verwenden möchten, ist eine statische Bibliothek ein passender Ort, um diese Funktionen (in übersetzter Form) aufzubewahren.

Win32-Konsolenanwendung

Das ist der Typ einer Windows-Anwendung, mit der Sie sich im ersten Teil dieses Sonderheftes hauptsächlich auseinandersetzen.



im Laufe dieses Beitrages Änderungen vornehmen. Die Dateien `stdafx.h` und `stdafx.cpp` dienen ausschließlich der Beschleunigung des Übersetzungsvorgangs – Sie können diese Dateien zunächst ignorieren. Die Datei `001.wsp` ist die *Workspace*-Datei mit den Einstellungen Ihres aktuellen Arbeitsbereiches und die Datei `001.dsp` ist die *Projekt*-Datei, die die Compiler- und Linkereinstellungen beinhaltet und in der außerdem aufgeführt ist, welche Dateien alle am Projekt beteiligt sind. Beide sind nicht dafür gedacht, bearbeitet zu werden, diese Dateien werden nur von VC++ intern benötigt. Mehr Informationen zu Arbeitsbereichen und Projekten finden Sie im Kasten "Projekte, Arbeitsflächen und Projekteinstellungen".

Am Bildschirm sehen Sie außerdem, dass ein neues Fenster geöffnet wurde: Links am Rand gibt es nun das *Arbeitsbereich*-Fenster, in dem Sie nicht nur die Einstellungen für Compiler und Linker verändern können, sondern auch alle am Projekt beteiligten Dateien finden. Im Reiter (Achtung: Die Reiter sind hier unten am Fenster angebracht) *Dateien* finden Sie eine dem

Explorer nachempfundene Ansicht, in der alle Dateien aufgelistet werden. Dabei wird zwischen *Quellcode*-, *Header*- und *Resource*-Dateien unterschieden.

Quellcode-Dateien sind die Dateien, die den tatsächlichen Programmcode enthalten. Header-Dateien nehmen verschiedene Informationen auf, die momentan noch nicht von Interesse sind, und Resource-Dateien enthalten Ressourcen, wie zum Beispiel Bilder oder Dialogboxen, also Elemente, die Sie für die Konsolen-Anwendung zunächst nicht benötigen.

Soweit so gut: Sie haben bis jetzt zwar noch keine einzige Zeile Quellcode programmiert, aber ein voll funktionsfähiger Quellcode ist dennoch vorhanden. Dieser liegt in Form der von VC++ erzeugten Datei `001.cpp` vor. Damit ist das ganze Projekt eigentlich mehr oder weniger fertig – Sie können es tatsächlich bereits übersetzen und starten.

■ Ein erster Durchlauf

Das geht mit den Befehlen aus dem Menü *Erstellen*. Wählen Sie hier zunächst einmal den Befehl *001.exe erstellen*, den Sie alternativ mit der Taste

[F7] erreichen können. Das Projekt wird dann "gebaut". Das geht in mehreren Stufen vor sich. Zunächst werden alle beteiligten Quellcode-Dateien (`001.cpp` und `stdafx.cpp`) vom

Compiler übersetzt. Dann werden die daraus resultierenden Dateien vom Linker zusammengebunden – und als Resultat kommt eine *.EXE*-Datei heraus.

Während das alles passiert werden entsprechende Meldungen unten im Ausgabefenster von VC++ angezeigt: Auch dieses Fenster öffnet sich automatisch.

Ist der Linker fertig, können Sie das Programm im Debugger starten. Wählen Sie dazu den Befehl *Debug starten* -> *ausführen* aus dem Menü erstellen, oder drücken Sie alternativ die Taste [F5]. Diese Taste sollten Sie sich merken, denn Sie werden sie in Zukunft sehr oft benötigen. Um genau zu sein, hätten Sie sich den separaten Übersetzungsschritt mit der Taste [F7] zuvor sparen können, denn VC++ überprüft jedes Mal vor dem Ausführen ob Quellcode übersetzt werden muss und erledigt das dann automatisch: Ein separater Übersetzungsschritt, den Sie manuell anwerfen, ist also eigentlich nicht notwendig.

Wie auch immer: Nachdem Sie [F5] gedrückt haben, wird das Programm gestartet. Sie sehen das in Form eines Konsolenfensters, das sich kurzfristig öffnet, dann den Text "Hallo Welt" anzeigt und direkt daraufhin wieder verschwindet: Herzlichen Glückwunsch – Sie haben soeben Ihr erstes eigenes Programmausgeführt!

Das Ganze ist aber natürlich ein Ereignis mit kurzer Lebenszeit: Schön wäre es ja schon, wenn man das Fenster ein wenig länger sehen könnte. Dazu muss aber etwas am Programmcode geändert werden – und dazu müssen Sie diesen überhaupt erst einmal begutachten. Genau das passiert im folgenden Schritt.

Dazu müssen Sie zunächst einmal den Quellcode des Programms in ein Editor-Fenster laden. Das geht am einfachsten, indem Sie im Arbeitsbereich unter *Dateien* den Ast für die Quellcode-Dateien öffnen und dann auf `001.cpp` doppelklicken. VC++ öffnet ein Fenster mit dem Quellcode Ihres Programms, und der sieht in etwa wie in Listing 2 auf der nächsten Seite aus.

■ Das Programm als Quellcode

Was Sie hier sehen sind 7 Zeilen CPP-Quellcode – hier zunächst nur eine ungefähre Beschreibung, was die Zeilen tun: Im nächsten Beitrag dieses Sonderheftes werden Sie viel tiefer in CPP eingeführt.



WIRD DAS PROGRAMM GESTARTET, öffnet sich ein Konsolenfenster und der Text "HalloWelt" wird darin angezeigt.

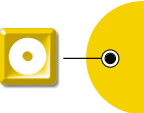
LISTING 1

```
// 001.cpp : Definiert den Einsprungpunkt für die Konsolenanwendung.
#include "stdafx.h"
int main(int argc, char* argv[])
{
    printf("Hallo Welt!\n");
    return 0;
}
```

TO C OR CPP – DAS IST DIE FRAGE

Mit VC++ können Sie sowohl "C"- als auch "CPP"-Programme schreiben, denn der Compiler von VC++ ist in der Lage, Quellcodes von beiden Sprachen zu übersetzen. Das kann manchmal etwas verwirrend sein, denn die Syntax der beiden Sprachen ist zwar sehr ähnlich, aber im Detail dann doch grundverschieden. Damit der Compiler weiß, in welcher Sprache ein Quellcode vorliegt, bzw. auf welche Art ein Quellcode übersetzt werden soll, unterscheidet er zwei Datei-Erweiterungen. Dateien mit der Erweiterung *.C*

werden als "C"-Programme angesehen und entsprechend übersetzt, während Dateien mit der Erweiterung *.CPP* als CPP-Programme übersetzt werden. Am einfachsten ist es allerdings, wenn Sie einfach immer CPP als Dateierweiterung verwenden, denn auch wenn Sie eigentlich ein "C"-Programm schreiben wollen, kann es nicht schaden, dieses mit dem "C" Subset von CPP zu programmieren – auf diese Weise macht man ganz sicher weniger Fehler.



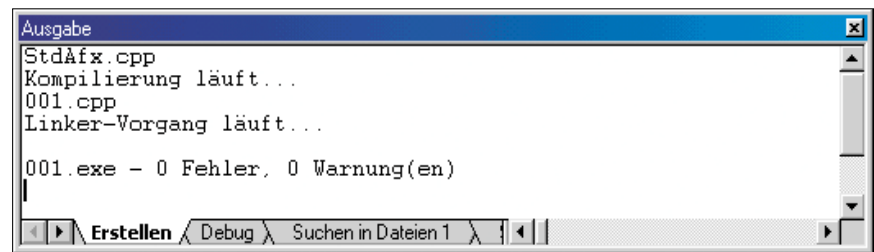
Die erste Zeile enthält einen Kommentar. Kommentare sind Textzeilen innerhalb des Quellcodes eines Programms, die nicht zum eigentlichen Programm gehören. Diese Teile werden nicht vom Compiler bei der Erstellung des Programms berücksichtigt – das bedeutet, Sie können in Kommentaren jeden Text unterbringen, der Ihnen gerade einfällt. Sinnvollerweise bringen Sie dort solche Texte unter, die die Datei oder das Programm beschreiben.

Die nächste Zeile ist ein Präprozessor-Statement. Solche Statements erkennen Sie immer an einer einleitenden Raute (#). Der Sinn und die Funktionsweise des Präprozessors wird an einer späteren Stelle beschrieben – für den Anfang reicht es zu wissen, dass Sie genau diese Zeile fast immer in den folgenden Beispielen finden werden.

Dann kommt eine Zeile, die eine Funktion einleitet – und zwar eine ganz bestimmte: die *main*-Funktion. Ihr Programm kann (und wird später auch) aus einer Vielzahl von Funktionen bestehen,

die sich gegenseitig aufrufen. Der Beginn des Programms kann aber natürlich nur an einer Stelle sein – irgendwo muss es ja schließlich anfangen.

Dieser Anfang wird durch eine ganz bestimmte Funktion definiert, und zwar durch die *main*-Funktion. Mit anderen Worten: Sie werden in praktisch



DAS AUSGABEFENSTER befindet sich normalerweise am unteren Fensterrand der IDE. Sie können es aber auch an eine andere Position verschieben.

LISTING 2

```
// 001.cpp : Definiert den Einsprungpunkt für die Konsolenanwendung.
#include "stdafx.h"
#include <conio.h>
int main(int argc, char* argv[])
{
    printf("Hallo Welt!\n");
    getch();
    return 0;
}
```




allen Programmen genau diese Funktion auf genau die gleiche Weise dargestellt sehen – es handelt sich hier immer um den Anfang des Programms.

Dann kommt ein Quellcode-Block, der in geschweiften Klammern ({) eingefasst ist. Diese Klammern dienen an dieser Stelle dazu, festzulegen, wo der Funktionskörper der *main()*-Funktion beginnt und wo er endet. Im Körper enthalten ist ein Funktionsaufruf (*printf()*) und ein einzelnes CPP-Statement (*return*).

Der Aufruf der *printf()*-Funktion dient dabei dazu, den Text auszugeben, der der Funktion als Parameter übergeben wurde. In diesem Fall ist es also der Text "Hallo Welt". Mit dem *return*-Statement wird die aktuelle Funktion verlassen. In diesem Fall handelt es sich dabei um die *main*-Funktion – und das bedeutet, dass mit dem *return*-Statement auch Ihr Programm beendet wird. Dabei hat *return* ebenfalls einen Parameter und zwar den Wert 0.

Der Parameter für *return* bestimmt, welcher Wert den aufrufenden Funktionen zurückgeliefert wird – was diese dann damit tut ist deren Sache. Im Falle der *main*-Funktion ist es so, dass dieser Rückgabewert an das Betriebssystem weitergegeben wird: Wenn Sie Ihr Programm nicht vom Debugger aus, sondern per Batch auf der Kommandozeile aufrufen, könnten Sie eine spezielle Environment-Variable auf den von *main()* per *return* zurückgelieferten Wert prüfen.

■ Eine Interaktion

Soweit so gut: Nun soll aber noch eine erste Veränderung am Programm vorgenommen werden. Fügen Sie dazu die zwei Zeilen in den Quelltext ein, sodass dieser danach das folgende Aussehen hat:

Der wichtigere Teil ist dabei der Aufruf der Funktion *getch()* (Der Name steht für "GET CCharacter" – also etwa



"hole Zeichen"). Wenn Sie dann das Programm mit der Taste [F5] neu übersetzen und starten lassen, wartet das Fenster mit der "Hallo Welt"-Anzeige so lange, bis Sie eine Taste drücken: Erst dann wird es geschlossen.

In diesem Teil des Sonderheftes haben Sie Ihr erstes CPP-Programm geschrieben, verändert und gestartet: Im nächsten Teil erfahren Sie, wie die einzelnen Befehle im Quelltext dabei zusammenhängen, wie C und C++ grundsätzlich funktionieren und wo die Unterschiede sind. UR

ERZEUGTE DATEIEN

Wenn Sie ein VC++-Projekt angelegt und übersetzt haben, entstehen beim Übersetzen des Projektes erneut Dateien. Dabei handelt es sich um die folgenden:

In Ihrem Projektverzeichnis werden die Dateien *PROJEKTNAME.opt* und *PROJEKTNAME.plg* angelegt. In der *opt*-Datei führt VC++ interne Informationen mit – Sie können diese Datei einfach ignorieren. Die *plg*-Datei enthält ein Protokoll des Übersetzungsvorganges im HTML Format: Auch diese Datei wird nicht wirklich verwendet, Sie kann aber hilfreich sein, wenn Sie Support-Anfragen stellen und nach Ihren Compiler-Einstellungen gefragt werden.

Dann legt VC++ je nach Art des Builds – also in Abhängigkeit davon, ob Sie ein Release oder ein Debug-Build erzeugt haben – noch

ein Unterverzeichnis "Debug" bzw. "Release" an. Darin werden die vom Compiler und Linker erzeugten Dateien abgelegt. Beim Erzeugen des ersten Projekts (001) legt VC++ die folgenden Dateien an:

- 001.exe
- 001.ilnk
- 001.obj
- 001.pch
- 001.pdb
- stdafx.obj
- vc60.idb
- vc60.pdb

An den Endungen der Dateien können Sie leicht erkennen, wozu Sie dienen. Die *.exe*-Datei ist dabei natürlich das ausführbare

Programm – also das Endprodukt des Übersetzungsvorgangs. Die *.ilnk*-Datei (ILK = Incremental Link) enthält Informationen, mit denen der Link-Vorgang beschleunigt wird. Die Datei *001.obj* ist das Resultat des Übersetzungsvorgangs der Quelldatei *001.cpp* – Sie finden für jede *.cpp*- oder *.c*-Datei in Ihrem Projekt auch eine solche Objektdatei (*obj* = Object), daher resultiert auch die Datei *stdafx.obj*. Die beiden *.pdb* (*pdb* = Program DataBase) Dateien enthalten Informationen über Ihr Programm. Um genau zu sein, sind dies die Informationen, die der Debugger benötigt, um Ihnen seine Funktionen beim Debuggen Ihres Programms zur Verfügung zu stellen.

PROJEKTE, ARBEITSFLÄCHEN UND PROJEKTEINSTELLUNGEN

Bei VC++ wird zwischen Arbeitsflächen und Projekten unterschieden. Mit Hilfe eines Projektes wird ein bestimmtes Programm erzeugt. Wollen Sie zum Beispiel eine Win32-Konsolenanwendung programmieren, benötigen Sie dazu ein Projekt. Dieses Projekt nimmt dann alle benötigten Quellcodes sowie die Compiler- und Linkereinstellungen, die Sie dafür vorgenommen haben auf.

Nun ist es aber so, dass es durchaus auch Programme geben kann, bei denen es praktisch ist, die Entwicklung in mehrere Projekte zu unterteilen: Zum Beispiel könnte ein Projekt für die Erzeugung einer bestimmten *.EXE*-Datei zuständig sein,

während weitere Projekte nur *.DLL*-Dateien erzeugen, die aber von der *.EXE*-Datei verwendet werden. Mehrere Projekte werden bei VC++ dann immer in einem "Arbeitsbereich" zusammengefasst. Dabei ist der Arbeitsbereich aber nicht optional – auch wenn Sie immer nur an einem Projekt arbeiten, legt VC++ dafür einen Arbeitsbereich an.

In den Projekteinstellungen wird von Haus aus zwischen zwei Sorten an Zielen unterschieden: Dabei handelt es sich um Debug- und Release-Versionen des gleichen Projekts. Dabei geht es um Folgendes: Damit Sie den Debugger benutzen können, um Ihr Programm zu untersuchen, müssen zusätz-

liche Informationen im Programm eingebettet werden – Informationen, die normalerweise nicht notwendig sind. Ferner darf der Optimizer das Programm nicht anfassen, denn sonst kommt der Debugger durcheinander. Das bedeutet, dass Sie Ihr Programm fürs Debuggen anders übersetzen müssen, als für eine "normal" zu betreibende Version. Genau das tun die Einstellungen in den Projektoptionen: Der eine Satz Einstellungen kümmert sich darum, dass eine Version des Programms erstellt wird, mit der der Debugger etwas anfangen kann, der andere Satz erzeugt ein Programm, das möglichst klein und schnell ist – aber eben nicht auf Quellcode-Niveau debugged werden kann.