Uwe Probst

Objektorientiertes Programmieren für Ingenieure

Anwendungen und Beispiele in C++

HANSER

Probst

Objektorientiertes Programmieren für Ingenieure

Uwe Probst

Objektorientiertes Programmieren für Ingenieure

Anwendungen und Beispiele in C++

Mit 105 Beispielen, 85 Bildern, 42 Listings, 15 Tabellen und 48 Übungen

Fachbuchverlag Leipzig
im Carl Hanser Verlag

Prof. Dr.-Ing. Uwe Probst
Technische Hochschule Mittelhessen

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN: 978-3-446-44234-4
E-Book-ISBN: 978-3-446-44178-1

Internet: http://www.hanser-fachbuch.de

Lektorat: Franziska Jacob, M.A.
Herstellung: Dipl.-Ing. (FH) Franziska Kaufmann
Satz: Kösel, Krugzell
Coverconcept: Marc Müller-Bremer, www.rebranding.de, München
Coverrealisierung: Stephan Rönigk
Druck und Bindung: Pustet, Regensburg
Printed in Germany

**»Der Weltuntergang steht bevor,
aber nicht so, wie Sie denken.
Dieser Krieg jagt nicht alles in die Luft,
sondern schaltet alles ab.«**

Tom DeMarco
Als auf der Welt das Licht ausging

ca. 560 Seiten. Hardcover
ca. € 19,99 [D] / € 20,60 [A] / sFr 28,90
ISBN 978-3-446-43960-3
Erscheint im November 2014

Hier klicken zur Leseprobe

**Sie möchten mehr über Tom DeMarco und seine Bücher erfahren.
Einfach reinklicken unter www.hanser-fachbuch.de/special/demarco**

Vorwort

Inzwischen sind im Zuge der Umsetzung des Bologna-Prozesses nahezu alle Diplom-Studiengänge auf die neuen Bachelor- und Masterprogramme umgestellt worden. Durch die Neuordnung wurden die Präsenzphasen an den Hochschulen gekürzt und Studierende zu mehr Eigenarbeit veranlasst. Diese Eigenarbeit anhand von Beispielen und überschaubaren Übungsaufgaben zielgerichtet zu strukturieren sowie mit einfach handhabbaren Beispielen zu unterstützen, ist ein wesentliches Ziel dieses Buches.

Die Inhalte basieren auf der Vorlesung „Softwareentwicklung“, die ich seit 2003 für Studierende der Elektrotechnik an der Technischen Hochschule Mittelhessen in dieser Form anbiete. Neben vielen Beispielen enthält das Buch eine Fülle von Übungsaufgaben mit ausführlichen Lösungsvorschlägen. Wesentliche Teile der Lösungsvorschläge sind jeweils am Ende des zugehörigen Hauptkapitels abgedruckt. Zudem stehen die vollständigen Projekte für Visual Studio 2010 auf der Seite http://www.oop-fuer-ingenieure.de.vu/ zum Herunterladen zur Verfügung.

Dieses Buch richtet sich gleichermaßen an Studierende und Mitarbeiter der Elektrotechnik an Universitäten und Fachhochschulen sowie an Ingenieure in der Praxis, die das objektorientierte Programmieren und die zugrunde liegenden Konzepte anhand von elektrotechnischen Fragestellungen erlernen und vertiefen wollen.

Ich danke allen an dieser Arbeit beteiligten Studenten und Mitarbeitern der Technischen Hochschule Mittelhessen, insbesondere Dipl. Ing. (FH) Johannes Friedrich für seinen engagierten Einsatz bei der Umsetzung der Inhalte in konkreten Lehrveranstaltungen. Ein besonderer Dank gebührt meinem Sohn Malte, der das gesamte Manuskript Korrektur gelesen und dank seines überragenden Sprachgefühls an entscheidenden Stellen in eine gute Form gebracht hat.

Gießen, im September 2014 *Uwe Probst*

URL der Internetseite mit den Applets zum Buch:

http://www.oop-fuer-ingenieure.de.vu/

Inhalt

1 Einleitung

Viele Maschinen und Anlagen im Bereich der industriellen Fertigung weisen einen hohen Anteil an Software auf. Die dort eingesetzten Programme bilden einen wesentlichen Teil des Hersteller-Know-hows und sind eng mit dem elektrischen bzw. mechatronischen Gesamtsystem verknüpft. Der Konzeption, Erstellung und Pflege dieser Programme kommt damit eine hohe Bedeutung zu. Praktisch alle Ingenieure und Naturwissenschaftler, die mit der Entwicklung und dem Vertrieb technischer Anlagen befasst sind, müssen daher ausreichende Programmierkenntnisse besitzen. Dies gilt auch dann, wenn sie nicht selbst programmieren, sondern Entwicklungsprozesse vorwiegend koordinieren. In diesem Fall sind grundlegende Kenntnisse der Softwareentwicklungsprozesse und der ihnen zugrunde liegenden Methoden und Konzepte unerlässlich.

Dieses Buch richtet sich daher an Studierende der Ingenieur- und Naturwissenschaften und an Absolventen, die bereits im Berufsleben stehen. Nach der Einleitung ist der Einstieg – abhängig von den individuellen Vorkenntnissen – an unterschiedlichen Stellen sinnvoll. Lesenden, die vorwiegend Erfahrungen mit prozeduraler Programmierung haben, wird ein Beginn mit Kapitel 2 empfohlen. Sofern bereits grundlegende Kenntnisse der objektorientierten Programmierung vorhanden sind, können die Kapitel 2 und 3 zunächst übersprungen und mit Kapitel 4 begonnen werden.

In Kapitel 2 werden wichtige nicht objektorientierte Erweiterungen von C zu C++ erläutert. Hierunter fallen sowohl Neuerungen für Funktionen unter den Stichworten variable Parameterliste, Überladen, Ausnahmebehandlung (Exceptions) und *inline* als auch der neue Datentyp *bool* sowie Referenzen, die für den effizienten Datenaustausch zwischen Funktionen große Bedeutung gewonnen haben.

Ein wesentliches Ziel der modernen objektorientierten Programmierung (OOP) ist es, eine gute Wiederverwendbarkeit bereits vorhandener Programmteile zu erreichen und zu unterstützen. Dies erfordert die Trennung des Programmcodes in *Schnittstelle* und *Implementierung*. Hauptthema in Kapitel 3 ist daher die Erläuterung der beiden wichtigen OOP-Paradigmen Geheimnisprinzip und Vererbung.

Kapitel 4 ist der dynamischen Speicherverwaltung gewidmet. Sie ermöglicht es, zur Programmlaufzeit eine beliebige Anzahl neuer Objekte anzulegen und ist damit essentiell für viele Anwendungen. Neben der Beschreibung der Vorgehensweise werden Datenstrukturen besprochen, die den Umgang mit solchen Objekten erleichtern. In diesem Zusammenhang wird auch das Thema Vielgestaltigkeit (Polymorphie) an Beispielen erörtert.

Um das Vorgehen zu strukturieren, wird die Softwareentwicklung in einzelne Phasen gegliedert. Zur Beschreibung des Ablaufs werden sogenannte Vorgehensmodelle verwendet, die die zeitliche Abfolge der einzelnen Phasen festlegen.

Moderne, leistungsfähige Softwareprodukte und die dafür erforderlichen Entwicklungsprozesse müssen einer ganzen Reihe von Anforderungen gerecht werden, um technisch und ökonomisch wettbewerbsfähig zu sein. Betrachtet man den Aufbau moderner Anwendungen eingehend, stellt man fest, dass vergleichbare Produkte oft einen ähnlichen Aufbau haben, der sich bewährt hat. So sind PKWs meist vierrädrig, haben zwischen 5 und 7 Sitze und eine Motorleistung bis zu 140 kW. Nahezu alle Einfamilienhäuser weisen – unabhängig vom jeweiligen Architektenentwurf – neben Wohn-, Ess- und Schlafzimmer noch zwei bis drei Kinderzimmer, Küche und Bad auf. Auf die Softwareentwicklung werden diese Prinzipien unter dem Stichwort Entwurfsmuster übertragen. Hierbei handelt es sich um etwa 25 unterschiedliche Softwarearchitekturen. Jede dieser Lösungen ist auf ganz bestimmte Anwendungen zugeschnitten. Unter dem Stichwort Softwaretechnik werden verschiedene Vorgehensmodelle und einige Entwurfsmuster in Kapitel 5 besprochen.

Um die Anwenderwünsche während der Analysephase eines Softwareprojekts möglichst präzise zu erfassen, ist eine exakte Beschreibung der zu implementierenden Module unerlässlich. Sie stellt gewissermaßen das Fundament des späteren Programms bereit. In der Praxis verwendet man hierzu die moderne Beschreibungssprache UML, die anhand von Beispielen erläutert wird. Dies erfolgt nicht in einem separaten Kapitel, sondern immer dann, wenn neue Sprachelemente eingeführt werden. Auf diese Weise werden innerhalb der einzelnen Kapitel wesentliche Diagrammtypen der UML vorgestellt und exemplarisch angewendet.

Mit Hilfe der prozeduralen Programmierung können die wesentlichen genannten Forderungen aufgrund systematischer Einschränkungen nicht konsequent umgesetzt werden. Möglich wird dies mit dem objektorientierten Ansatz, der sowohl beim Erfassen der Anforderungen als auch in der Entwurfs- und Implementierungsphase zum Einsatz kommt. Ein unverzichtbares Hilfsmittel stellen fertige objektorientierte Bibliotheken dar, die häufig auf Vorlagen und Schablonen basieren. Beispiele für solche Bibliotheken sind die C++-Standardbibliothek (STL) sowie die Microsoft Foundation Classes (MFC), die beide in Kapitel 6 besprochen werden. Letztere bildet eine mögliche Grundlage zur Erstellung grafischer Oberflächen.

Den Abschluss bilden ingenieurwissenschaftliche Programmierbeispiele in Kapitel 7.

Die wesentlichen Sprachelemente der Programmiersprache C werden für das Verständnis des Buches vorausgesetzt (z. B. Arrays, Zeiger, Parameterübergabe bei Funktionen, Schleifen und Kontrollstrukturen). Alle Beispielprogramme wurden mit Visual Studio 2010 entwickelt und getestet. Auf der Webseite zum Buch können diese Beispiele einzeln heruntergeladen werden. Grundsätzlich sollten die Quellcodes der *.cpp-Dateien – mit Ausnahme der Beispiele in den Kapiteln 6 und 7, die herstellerspezifische Bibliotheken nutzen – auch mit anderen Compilern erfolgreich übersetzt werden können.

Die dargestellten Beispiele sollen jeweils gewisse Aspekte verdeutlichen und erheben nicht den Anspruch von Robustheit und Zuverlässigkeit. Man kann alles anders und besser machen.

2 Von C zu C++

Im Vergleich zur Sprache C enthält C++ eine ganze Reihe von Verbesserungen und Weiterentwicklungen, die nicht unmittelbar mit den objektorientierten Programmkonzepten zusammenhängen. Eine Auswahl dieser Erweiterungen wird nachfolgend besprochen.

2.1 Neues zu Funktionen

Lernziele
Der/die Lernende

- nutzt eine variable Parameterliste zur Vereinfachung der Schnittstelle,
- wendet die Technik des Überladens auf Funktionen an,
- erkennt die Vorteile von Funktionsschablonen.

2.1.1 Funktionen mit variabler Parameterliste

In C++ ist es möglich, dass man für einige der Funktionsparameter Standardwerte vorgibt. Werden die Parameter beim Aufruf nicht explizit vorgegeben, verwendet die Funktion automatisch die Standardwerte. Parameter dieser Art müssen am Ende der Parameterliste platziert werden:

```
void inkrement(int *x, int step=1) {
    *x += step;
}
void main(void) {
    int wert = 0;
    inkrement (&wert, 2);          // individuelle Angabe der Schrittweite: 2
    inkrement(&wert);              // Schrittweite nicht angegeben: 1
}
```

Die Funktion *inkrement(int *x, int step=1)* verwendet zwei Parameter. Der Parameter *step* wird mit dem Wert 1 vorbelegt. Der erste Aufruf von *inkrement* setzt *step* auf den Wert 2. Beim zweiten Aufruf wird dagegen nur der Zeiger übergeben. Da *step* nicht spezifiziert wurde, wird hier mit dem Standardwert 1 gearbeitet.

Standardwerte für Funktionsparameter werden mittels Zuweisungsoperator („=“) den formalen Parametern nachgestellt. Dabei gilt, dass nach einem Parameter mit Standardwert nur noch Parameter folgen dürfen, welchen ebenfalls ein Standardwert zugewiesen ist.