

Uwe Probst

# Objektorientiertes Programmieren für Ingenieure

Anwendungen und Beispiele in C++



HANSER



**Blieben Sie auf dem Laufenden!**

Hanser Newsletter informieren Sie regelmäßig über neue Bücher und Termine aus den verschiedenen Bereichen der Technik. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter

**[www.hanser-fachbuch.de/newsletter](http://www.hanser-fachbuch.de/newsletter)**



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



Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autoren und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

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

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung – mit Ausnahme der in den §§ 53, 54 URG genannten Sonderfälle –, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2015 Carl Hanser Verlag München

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](http://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](http://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

<b>1</b>	<b>Einleitung .....</b>	<b>11</b>
<b>2</b>	<b>Von C zu C++ .....</b>	<b>13</b>
2.1	Neues zu Funktionen .....	13
2.1.1	Funktionen mit variabler Parameterliste .....	13
2.1.2	Überladen .....	14
2.1.3	Inline-Funktionen .....	15
2.2	Referenzen .....	16
2.2.1	Definition .....	16
2.2.2	Referenzen und Funktionen .....	18
2.3	Datentyp <i>bool</i> .....	20
2.4	Namensbereiche .....	20
2.5	Ein- und Ausgabe .....	22
2.5.1	Standardausgabe .....	22
2.5.2	Standardeingabe .....	25
2.5.3	Lesen und Schreiben von Dateien .....	25
2.5.4	Ausgabe selbst definierter Datentypen .....	27
2.6	Exceptions und Fehlerbehandlung .....	28
2.6.1	Einführung .....	28
2.6.2	Ausnahmen .....	30
2.6.3	Verwendung vorhandener Exception-Klassen .....	32
2.7	Lösungen .....	37
<b>3</b>	<b>Klassen und Objekte .....</b>	<b>39</b>
3.1	Motivation zur Einführung der OOP .....	39
3.2	Von <i>struct</i> zu <i>class</i> .....	43
3.2.1	Klassendeklaration .....	43
3.2.2	Geheimnisprinzip: <i>private</i> und <i>public</i> .....	44
3.2.3	Objekte und Variablen .....	45
3.2.4	Konstruktor und Destruktor .....	45
3.2.5	Objekt: Instanz einer Klasse .....	50
3.2.6	Aufteilung in Dateien .....	53
3.2.7	Konstante Objekte .....	54
3.2.8	Hinweise zur Konstruktion von Methodenschnittstellen .....	55
3.2.9	<i>this</i> -Zeiger .....	56

3.2.10	Zusammengesetzte Klassen .....	57
3.2.11	Überladen von Operatoren .....	62
3.3	Vererbung .....	66
3.3.1	Gemeinsamkeiten und Unterschiede .....	66
3.3.2	Vererbungshierarchie .....	68
3.3.3	Ableitung und Zugriffsrechte .....	70
3.3.4	Initialisierung bei abgeleiteten Klassen .....	74
3.3.5	Abstrakte Klassen .....	74
3.4	Lösungen .....	75
<b>4</b>	<b>Dynamische Speicherverwaltung .....</b>	<b>81</b>
4.1	Dynamisches Anlegen von Objekten .....	81
4.2	Felder mit dynamisch änderbarer Länge .....	83
4.2.1	Einführende Beispiele .....	83
4.2.2	Unterschied zwischen tiefer und flacher Kopie .....	88
4.2.3	Initialisierung durch Kopie .....	90
4.3	Behälterklassen .....	91
4.3.1	Einfach verkettete Listen .....	91
4.3.2	Weitere Arten von Behälterklassen .....	99
4.4	Polymorphie am Beispiel von Listen .....	100
4.4.1	Einführung .....	100
4.4.2	Einfache Liste mit Bauelementen .....	101
4.4.3	Implizite Typumwandlungen in C++ .....	103
4.4.4	Polymorphie .....	105
4.5	Lösungen .....	108
<b>5</b>	<b>Techniken der Softwareentwicklung .....</b>	<b>113</b>
5.1	Grundlagen .....	114
5.1.1	Fünf Phasen der Softwareentwicklung .....	114
5.1.2	Vorgehensmodelle .....	115
5.2	Entwurfsmuster .....	120
5.2.1	Grundlagen von Entwurfsmustern .....	120
5.2.2	Strategie .....	122
5.2.3	Adapter .....	127
5.2.4	Beobachter .....	132
5.3	Lösungen .....	141
<b>6</b>	<b>Klassenbibliotheken .....</b>	<b>148</b>
6.1	Vorlagen, Schablonen, Templates .....	149
6.1.1	Makros .....	149
6.1.2	Templates .....	151
6.1.2.1	Funktionstemplates .....	151
6.1.2.2	Klassentemplates .....	154
6.2	Wichtige Bestandteile der C++-Standardbibliothek .....	157

6.3	Microsoft Foundation Classes (MFC) .....	162
6.3.1	Grafische Benutzeroberflächen .....	162
6.3.2	Grundlegendes zu Windows-Programmen .....	163
6.3.3	Aufbau der MFC .....	168
6.3.4	Programmierung eines einfachen Taschenrechners .....	172
6.3.5	Grafische Ausgabe .....	180
6.4	Lösungen .....	197

**7** **Beispielanwendungen** ..... **205**

7.1	Visualisierung von Messwerten .....	205
7.1.1	Anwendungsfälle .....	206
7.1.2	Analyse .....	207
7.1.3	Entwurf .....	208
7.2	Erstellen von Bode-Diagrammen .....	214
7.2.1	Analyse .....	214
7.2.2	Entwurf .....	215
7.3	Lösungen .....	218
7.4	Zusammenfassung .....	220

**Literatur** ..... **223**

**Index** ..... **225**



# 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 Programmaufzeit 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 vierrädig, 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 herstellereigene 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.