Objekttechnologie

Reihenherausgeber

Martin Schader, Mannheim

Springer-Verlag Berlin Heidelberg GmbH

Objekttechnologie

Martin Schader und Michael Rundshagen
Objektorientierte Systemanalyse
2. Auflage
IX, 241 Seiten. 1996

Martin Schader
Objektorientierte Datenbanken
Die C++-Anbindung des
ODMG-Standards
X, 219 Seiten. 1997

Günther Vinek
Objektorientierte Softwareentwicklung
mit Smalltalk
XII, 440 Seiten. 1997

Martin Schader und Stefan Kuhlins
Programmieren in C++
Einführung in den Sprachstandard
5. Auflage
XII, 386 Seiten. 1998

Martin Schader und Lars Schmidt-Thieme
Java
Einführung in die objektorientierte
Programmierung
XVII, 544 Seiten. 1998

Stefan Kuhlins
Martin Schader

Die C++-Standard-bibliothek

Einführung und Nachschlagewerk

Mit 77 Abbildungen
und 37 Tabellen

Springer

Dr. Stefan Kuhlins
Prof. Dr. Martin Schader
Universität Mannheim
Lehrstuhl für Wirtschaftsinformatik III
Schloß
D-68131 Mannheim

ISBN 978-3-540-65052-2

Die Deutsche Bibliothek – CIP-Einheitsaufnahme
Kuhlins, Stefan: Die C++-Standardbibliothek: Einführung und Nachschlagewerk / Stefan Kuhlins; Martin Schader.
(Objekttechnologie)
ISBN 978-3-540-65052-2 ISBN 978-3-662-06640-9 (eBook)
DOI 10.1007/978-3-662-06640-9

Ursprünglich erschienen bei Springer-Verlag Berlin Heidelberg New York 1999

Umschlaggestaltung: Struwe & Partner, Heidelberg

SPIN 10694940 43/2202-5 4 3 2 1 0 – Gedruckt auf säurefreiem Papier

Vorwort

Warum wird eine Standardbibliothek für C++ gebraucht? Programmentwickler benötigen immer wieder die gleichen Datenstrukturen wie z.B. dynamische Felder. Aufgrund verschiedener konkreter Anforderungen werden einzelne Implementationen aber i.d.R. sehr unterschiedlich ausfallen. Dadurch werden Programme anderer Entwickler schwer verständlich, und beim Wechsel der eingesetzten Bibliotheken entsteht erheblicher Lernaufwand. Durch eine Standardbibliothek lassen sich diese Probleme in den Griff bekommen. Um sich an die verschiedenen Anforderungen, unter denen eine Bibliothek zum Einsatz kommt, anpassen zu können, ist eine wichtige Voraussetzung für den Erfolg einer Standardbibliothek, daß sie flexibel und erweiterbar ist. Mit der *Standard Template Library* (STL), dem Kernstück der aktuellen C++-Standardbibliothek, wird dieses Ziel erreicht.

Darüber hinaus ist die STL äußerst effizient, so daß sich eine hervorragende Performance erreichen läßt. Allerdings setzt ein gewinnbringender Einsatz der STL ein tiefes Verständnis für das Design der STL voraus. Um die STL zu verstehen, genügt es nicht, lediglich eine Funktionsreferenz zu lesen. Deshalb haben wir in diesem Buch besonderen Wert darauf gelegt, die Konzepte und Funktionsweisen der STL zu erläutern. In diesem Zusammenhang ist vor allem Kapitel 2 hervorzuheben, in dem wir einen möglichen Entwicklungsweg für die STL aufzeigen.

Bei der Beschreibung der einzelnen Komponenten der STL – Container, Iteratoren, Algorithmen, Funktionsobjekte usw. – gehen wir auch auf Implementationsdetails ein, die vom Standard zwar nicht festgeschrieben sind, sich i.d.R. aber aufgrund von Vorgaben für die Komplexität automatisch ergeben. Das Wissen um Implementationsdetails unterstützt eine effiziente Benutzung der STL. Ferner demonstrieren zahlreiche kleine Anwendungsbeispiele den Gebrauch.

Im Schatten der STL existieren viele weitere interessante Komponenten der C++-Standardbibliothek, zu denen u.a. *Streams*, *Strings*, *Auto-Pointer*, *Bitsets* und komplexe Zahlen gehören. Auch für diese Klassen liefern wir die für den praktischen Einsatz notwendigen Informationen.

Unsere Ausführungen basieren auf dem aktuellen Entwurf für den C++-Standard. Die Arbeiten am C++-Standard, die vom *American National Standards Institute* (ANSI J16) und von der *International Standards Organization* (ISO WG21) vorangetrieben werden, sind gegenwärtig quasi abgeschlossen. Der aktuelle Stand wird durch den *Final Draft International Standard* (FDIS) vom November 1997 repräsentiert. Laut Zeitplan soll der endgültige Standard unter der Bezeichnung ISO/IEC 14882, *Standard for the C++ Programming Language*,

im Oktober 1998 erscheinen. Er soll sich vom FDIS lediglich durch die Korrektur von typographischen und Layoutfehlern unterscheiden.

Gegenüber der Urfassung der STL, die von *Alexander Stepanov* und *Meng Lee* bei *Hewlett Packard* als Technischer Report HPL-94-34 mit dem Titel *„The Standard Template Library"* im April 1994 erschienen ist, sind während der Standardisierung unzählige Änderungen vorgenommen worden. An Stellen, an denen es uns nützlich erschien, weisen wir auf frühere Besonderheiten hin, damit sich unsere Programmbeispiele an ältere Versionen der STL anpassen lassen. Es ist allerdings zu erwarten, daß die Compiler- und Bibliothekshersteller innerhalb kürzester Zeit den aktuellen C++-Standard umsetzen werden.

Da wir hier den C++-Standard beschreiben, sollten sich alle unsere Beispielprogramme mit jedem standardkonformen C++-Compiler erfolgreich übersetzen lassen. Zum Testen unserer Programmbeispiele haben wir die folgenden derzeit verfügbaren C++-Compiler eingesetzt (in alphabetischer Reihenfolge):

- Borland C++ unter Windows 95/NT
- EGCS unter Linux und Solaris
- GNU g++ unter Linux und Solaris
- Kuck & Associates, Inc. KAI C++ unter Linux und Solaris
- Microsoft Visual C++ unter Windows 95/NT

Jedes Beispielprogramm läßt sich mit mindestens einem dieser Compiler erfolgreich übersetzen. Darüber hinaus decken die im Text angegebenen Definitionen für Klassen, Funktionen usw. einen Großteil der STL ab. Insbesondere älteren Implementationen kann man damit auf die „Sprünge" helfen, so daß man seine Programme gemäß dem aktuellen Standard schreiben kann und späterer Umstellungsaufwand entfällt.

Die C++-Standardbibliothek reizt die Programmiersprache C++ voll aus. Das geht sogar so weit, daß die Sprache für die Bibliothek erweitert wurde; *Member-Templates* sind dafür ein Beispiel (siehe Seite 65). Um die in diesem Buch dargestellten Themen verstehen zu können, muß man über fundierte C++-Kenntnisse verfügen. Zum Erlernen der Sprache C++ und als Nachschlagewerk empfehlen wir gerne unser Buch *„Programmieren in C++"*, das ebenfalls im Springer-Verlag erschienen ist.

Wir gehen davon aus, daß das Buch in der vorgegebenen Reihenfolge durchgearbeitet wird. Um langweilende Wiederholungen zu vermeiden, werden z.B. Elementfunktionen von Containerklassen nur beim ersten Auftreten im Text ausführlich und mit Beispielen erläutert.

Jeweils am Kapitelende haben wir einige Aufgaben zusammengestellt. Lösungsvorschläge befinden sich in Kapitel 15. Die meisten Aufgaben beschäftigen sich mit Themen, die über das im Text gesagte hinausgehen. Dementsprechend ist der Schwierigkeitsgrad recht hoch, und ein Blick auf die Lösungen lohnt sich.

Der Sourcecode aller mit dem Diskettensymbol 🖫 gekennzeichneten Programme und Lösungen ist mittels *Anonymous ftp* auf dem Server ftp.wifo.uni-mannheim.de im Unterverzeichnis /pub/buecher/stlbuch zugänglich. Alternativ ist er gegen Einsendung einer MS-DOS-formatierten 3½"-Diskette mit ausreichend frankiertem Rückumschlag an die auf den Umschlagseiten angegebene Anschrift erhältlich.

Über Anregungen unserer Leserinnen und Leser an unsere Postanschrift oder als *Email* an stlbuch@wifo.uni-mannheim.de würden wir uns freuen. Wer uns einen Fehler (gleichgültig welcher Art) zuerst mitteilt, den erwähnen wir namentlich im *World Wide Web* auf unseren Seiten zum Buch unter http://www.wifo.uni-mannheim.de/stlbuch, wo wir auch aktuelle Informationen zur C++-Standardbibliothek und Hinweise zu speziellen Compilern zur Verfügung stellen.

Wir danken Michael Gröschel und Rainer Lienhart für die Durchsicht des Manuskripts. Außerdem gilt unser Dank dem Springer-Verlag für die, wie immer, sehr gute Zusammenarbeit.

Mannheim, im August 1998 *Stefan Kuhlins, Martin Schader*

Inhaltsverzeichnis