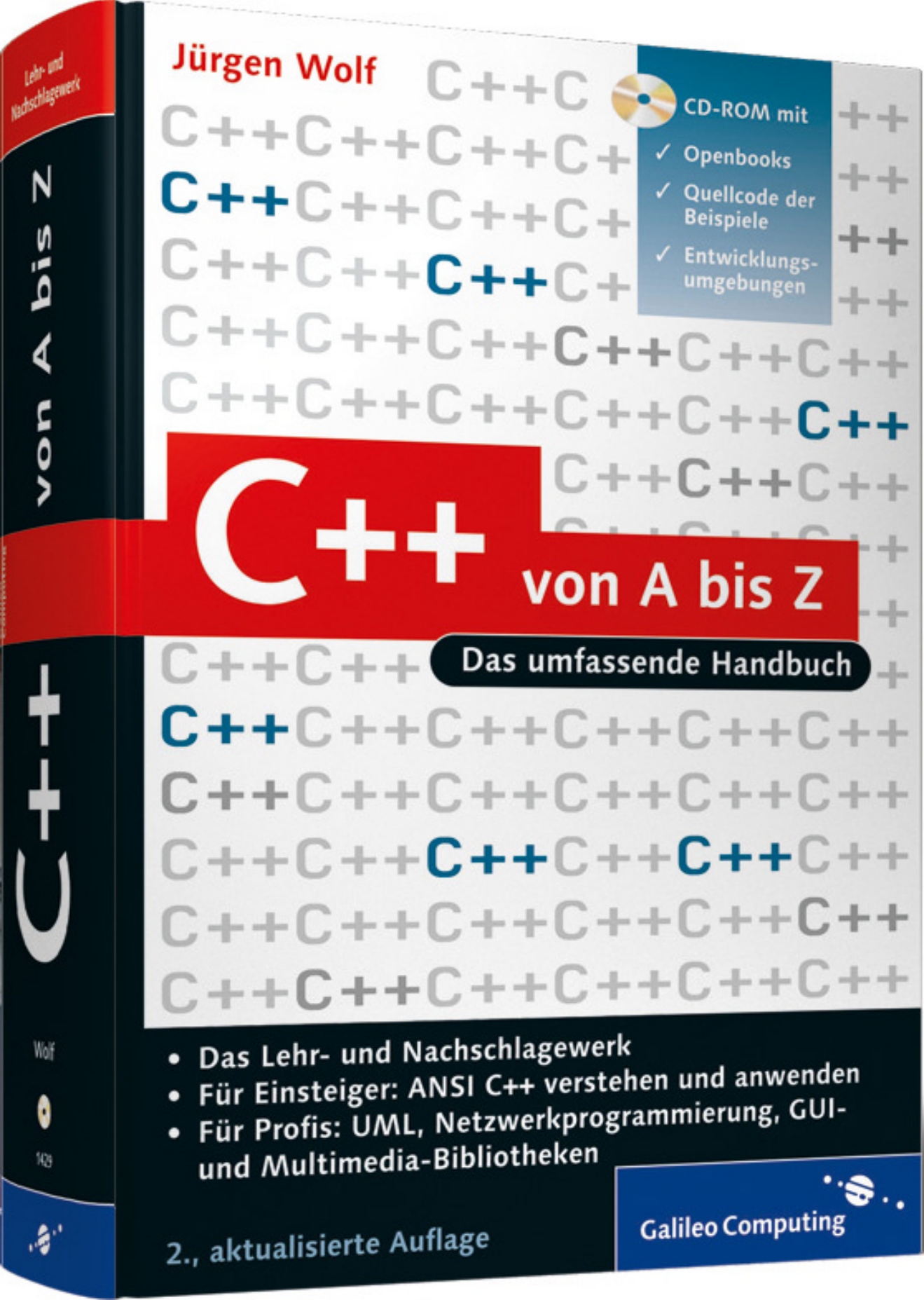
Jürgen Wolf
CD-ROM mit
✓ Openbooks
✓ Quellcode der Beispiele
✓ Entwicklungsumgebungen
C++ von A bis Z
Das umfassende Handbuch
• Das Lehr- und Nachschlagewerk
• Für Einsteiger: ANSI C++ verstehen und anwenden
• Für Profis: UML, Netzwerkprogrammierung, GUI- und Multimedia-Bibliotheken
2., aktualisierte Auflage
Galileo Computing
Lehr- und Nachschlagewerk
C++ von A bis Z
Wolf
1429

Auf einen Blick

Inhalt

Dieses Kapitel geht auf die Grundlagen der C++-Programmierung bzw. auf die grundlegenden Themen der meisten Programmiersprachen überhaupt ein. Von Basisdatentypen, Standard-I/O-Streams, Konstanten, lexikalischen Elementen, Operatoren, Begrenzern, verschiedenen Kontrollstrukturen und Funktionen bis zum Präprozessor finden Sie hier vieles, was Ihnen in anderen Programmiersprachen recht ähnlich (oder fast gleich) begegnet. Hierbei soll auch kurz auf die Geschichte von C++ und auf die Frage eingegangen werden, wie man eigentlich ein Programm erstellt.

1 Grundlagen in C++

1.1 Die Entstehung von C++

Ursprünglich wurde C++ 1979 von Dr. Bjarne Stroustrup entwickelt, um Simulationsprojekte mit geringem Speicher- und Zeitbedarf zu programmieren. Auch hier lieferte (wie schon bei C) kein geringeres Betriebssystem als UNIX die Ursprungsplattform dieser Sprache. Stroustrup musste (damals noch bei den Bell Labs beschäftigt) den UNIX-Betriebssystemkern auf verteilte Programmierung hin analysieren. Für größere Projekte verwendete Stroustrup bisher die Sprachen Simula – die allerdings in der Praxis recht langsam bei der Ausführung ist – und BCPL, die zwar sehr schnell ist, aber sich für große Projekte nicht eignete.

Simula und BCPL

Simula gilt als Vorgänger von Smalltalk. Viele der mit Simula eingeführten Konzepte finden sich in modernen objektorientierten Programmiersprachen wieder.

BCPL (Kurzform für *Basic Combined Programming Language*) ist eine um 1967 von Martin Richards entwickelte kompilierte, systemnahe Programmiersprache, abgeleitet von der *Combined/Cambridge Programming Language* CPL. Es handelt sich um eine Sprache aus der ALGOL-Familie.

Homepage von Stroustrup

Wer mehr über Bjarne Stroustrup erfahren möchte, findet unter der URL *http://public.research.att.com/~bs/homepage.html* seine Homepage.

Stroustrup erweiterte die Sprache C um ein Klassenkonzept, wofür er die Sprache Simula-67 (mit der Bildung von Klassen, Vererbung und dem Entwurf virtueller Funktionen) und später dann Algol68 (Überladen von Operatoren; Deklarationen im Quelltext frei platzierbar) sowie Ada (Entwicklung von Templates, Ausnahmebehandlung) als Vorlage nahm. Heraus kam ein »C mit Klassen« (woraus etwas später auch C++ wurde). C wurde als Ursprung verwendet, weil diese Sprache schnellen Code erzeugte, einfach auf andere Plattformen zu portieren ist und fester Bestandteil von UNIX ist. Vor allem ist C auch eine weitverbreitete Sprache, wenn nicht sogar die am weitesten verbreitete Sprache überhaupt. Natürlich wurde auch weiterhin auf die Kompatibilität von C geachtet, damit auch in C entwickelte Programme in C++-Programmen liefen. Insgesamt wurde »C mit Klassen« zunächst um folgende Sprachelemente erweitert, auf die später noch eingegangen wird:

- Klassen
- Vererbung (ohne Polymorphismus)
- Konstruktoren, Destruktoren
- Funktionen
- Friend-Deklaration
- Typüberprüfung

Einige Zeit später, 1982, wurde aus »C mit Klassen« die Programmiersprache C++. Der Inkrementoperator ++ am Ende des C sollte darauf hinweisen, dass die Programmiersprache C++ aus der Programmiersprache C entstanden ist und erweitert wurde. Folgende neue Features sind dann gegenüber »C mit Klassen« hinzugekommen:

- virtuelle Funktionen
- Überladen von Funktionen
- Überladen von Operatoren
- Referenzen
- Konstanten
- veränderbare Freispeicherverwaltung
- verbesserte Typüberprüfung
- Kommentare mit // am Zeilenende anfügen (von BCPL)

1985 erschien dann die Version 2.0 von C++, die wiederum folgende Neuerungen enthielt:

- Mehrfachvererbung
- abstrakte Klassen

- statische und konstante Elementfunktionen
- Erweiterung des Schutzmodells um das Schlüsselwort `protected`

Relativ spät, 1991, fand das erste Treffen der ISO(*International Organisation for Standardization*)-Workgroup statt, um C++ zu standardisieren, was 1995 zu einem *Draft Standard* führte.

Draft Standard

Draft Standard ist die Vorstufe zum Standard. Das Protokoll hat die Analyse- und Testphase bestanden, kann jedoch noch modifiziert werden.

Allerdings dauerte es wiederum drei weitere Jahre, bis 1998 C++ dann endlich von der ISO genormt wurde (ISO/IEC 14882:1998). Erweitert wurde C++ in dieser Zeit um folgende Features:

- Templates
- Ausnahmebehandlung
- Namensräume
- neuartige Typumwandlung
- boolesche Typen

Natürlich entstanden während der Zeit der Weiterentwicklung von C++ auch eine Menge Standardbibliotheken wie bspw. die Stream-I/O-Bibliothek, die die bis dahin traditionellen C-Funktionen `printf()` und `scanf()` abgelöst haben. Ebenfalls eine gewaltige Standardbibliothek wurde von HP mit STL (Standard Template Library) hinzugefügt.

Im Jahre 2003 wurde dann die erste überarbeitete Version von ISO/IEC 14882:1998 verabschiedet und ISO/IEC 14882:2003 eingeführt. Allerdings stellt diese Revision lediglich eine Verbesserung von ISO/IEC 14882:1998 dar und keine »neue« Version. Eine neue Version von C++ (auch bekannt unter C++0x bzw. C++200x) soll angeblich noch in diesem Jahrzehnt erscheinen – zumindest deutet der Name dies an.

Seit Ende 2006 wurde als Termin für die Fertigstellung das Jahr 2009 erwähnt. Somit wurde aus C++0x die Abkürzung C++09. Ob der Termin tatsächlich eingehalten wird, steht natürlich wieder auf einem anderen Blatt. Und neben dem Einhalten des Termins müssen auch die Compiler der Hersteller entsprechend erneuert werden. Man kann nur hoffen, dass die Compiler-Hersteller den neuen C++09-Standard konsequent umsetzen werden (beim `g++`, denke ich, wird dies recht schnell gehen), so dass dann bis 2010 bzw. spätestens 2011 alle Compiler

auf den neuen Standard aufbauen. Aber wie gesagt, das ist lediglich Wunschdenken des Autors.

Zu einem der Top-Features im neuen C++09-Standard gehört ganz klar die Unterstützung von Threads. Ein Thema, das früher vom Standardisierungskomitee nie beachtet wurde, muss jetzt im Zeitalter von Mehrprozessorumgebungen einfach standardisiert werden. Der C++09-Standard wird auf jeden Fall eine eigene Bibliothek zur Unterstützung von Threads enthalten.

Auch die Programmbibliothek wird u. a. um reguläre Ausdrücke, Zufallsbibliothek, intelligente Zeiger, neue untergeordnete assoziative Container, Tupel, Werkzeuge für C++-Metaprogrammierung usw. erweitert. Viele dieser Erweiterungen sind Teil der Boost-Bibliothek und werden mit minimaler Veränderung übernommen. Interessant ist auch, dass der C99-Standard in einer für C++ geänderten Form enthalten sein wird.

Auch der Sprachkern soll im C++09-Standard verbessert bzw. vereinfacht werden. Neu hinzukommen werden Konzepte (engl.: *concepts*) zur Spracherweiterung.

[»] **Hinweis**

Weitere Informationen (Weblinks) zum künftigen C++09-Standard habe ich Ihnen auf der Buch-CD zusammengestellt.

1.1.1 Aufbau von C++

C++ ist im Gegensatz zu Sprachen wie Smalltalk oder Eiffel keine reine objektorientierte Sprache – genauer gesagt: Smalltalk bzw. Eiffel wurden ohne Wenn und Aber als objektorientierte Sprachen entwickelt. C++ hingegen entstand ja aus C – womit es sich hierbei um eine Programmiersprache mit objektorientierter Unterstützung handelt, was zum einen Nachteile, zum anderen aber auch Vorteile mit sich bringt (siehe »Stärken von C++« und »Schwächen von C++« weiter unten in diesem Abschnitt).

[»] **Hinweis**

Sofern Sie bereits mit C vertraut sind, können Sie die folgenden Zeilen als »Update« betrachten, das Sie darüber informiert, was mit C++ alles neu auf Sie (als C-Umsteiger) zukommt.

Erweiterungen von C

Neben der objektorientierten Unterstützung bietet C++ auch sprachliche Erweiterungen von C an. Zu den bekanntesten Erweiterungen gehören folgende Punkte:

- Inline-Funktionen
- Default-Argumente
- Referenztypen
- Überladen von Funktionen
- Überladen von Operatoren
- Templates
- Ausnahmebehandlung

Objektorientierte Unterstützung

Die objektorientierte Unterstützung von C++ enthält folgende Möglichkeiten:

- Klassen bilden
- Zugriff auf Daten und Elementfunktionen mit `public`-, `private`- oder `protected`-Spezifikationen steuern
- Klassen vererben (auch mehrfach)
- polymorphe Klassen bilden

Stärken von C++

Wie bereits erwähnt, hat C++, wie jede andere Sprache auch, einige »schwache« und »starke« Seiten. Zu den Stärken von C++ gehören folgende Punkte:

- maschinennahes Programmieren
- Erzeugung von hocheffizientem Code
- hohe Ausdrucksstärke und Flexibilität
- für umfangreiche Projekte geeignet
- sehr weite Verbreitung
- Keine Organisation (wie z. B. bei Java) hat hier ihre Finger mit im Spiel (die Standardisierung erfolgt durch die ISO).
- viele Möglichkeiten für die Metaprogrammierung
- C-kompatibel – dadurch kann das Programm, das in C erstellt wurde, weiterhin unverändert verwendet werden. Außerdem braucht sich ein C-Programmierer beim Umstieg nur mit den Erweiterungen und der objektorientierten Programmierung auseinanderzusetzen.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z