

2. Auflage

Ulrich BREYMANN

DER C++ PROGRAMMIERER

C++ LERNEN -
PROFESSIONELL ANWENDEN -
LÖSUNGEN NUTZEN



EXTRA: Mit kostenlosem E-Book



Auf DVD: Entwicklungsumgebung, Compiler,
weitere Tools, alle Beispiele des Buchs



Topaktuell: Entspricht dem neuen
ISO-C++-Standard

HANSER

Breymann



Der C++-Programmierer



Bleiben Sie einfach auf dem Laufenden:
www.hanser.de/newsletter

Sofort anmelden und Monat für Monat
die neuesten Infos und Updates erhalten.

Ulrich Breymann



Der C++-Programmierer

**C++ lernen -
Professionell anwenden -
Lösungen nutzen**

2., aktualisierte Auflage

HANSER

Prof. Dr. Ulrich Breymann lehrt Informatik an der Fakultät Elektrotechnik und Informatik
der Hochschule Bremen
Kontakt: breymann@hs-bremen.de

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. Autor 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 Autor 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.

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 – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2011 Carl Hanser Verlag München (www.hanser.de)

Lektorat: Margarete Metzger

Copy editing: Jürgen Dubau, Freiburg

Herstellung: Irene Weilhart

Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München

Umschlagrealisation: Stephan Rönigk

Datenbelichtung, Druck und Bindung: Kösel, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

ISBN 978-3-446-42691-7

E-Book-ISBN 978-3-446-42841-6

Inhalt

Vorwort	21
Teil I: Einführung in C++	25
1 Es geht los!	27
1.1 Historisches	27
1.2 Objektorientierte Programmierung	28
1.3 Compiler	31
1.4 Das erste Programm	31
1.4.1 Namenskonventionen	36
1.5 Integrierte Entwicklungsumgebungen	37
1.5.1 Code::Blocks	37
1.5.2 Eclipse	39
1.6 Einfache Datentypen und Operatoren	41
1.6.1 Ausdruck	42
1.6.2 Ganze Zahlen	42
1.6.3 Reelle Zahlen	47
1.6.4 Konstante	50
1.6.5 Zeichen	51
1.6.6 Logischer Datentyp bool	54
1.6.7 Referenzen	55
1.6.8 Regeln zum Bilden von Ausdrücken	56
1.6.9 Standard-Typumwandlungen	57

1.7	Gültigkeitsbereich und Sichtbarkeit	58
1.7.1	Namespace std	60
1.8	Kontrollstrukturen	61
1.8.1	Anweisungen	61
1.8.2	Sequenz (Reihung)	63
1.8.3	Auswahl (Selektion, Verzweigung)	63
1.8.4	Fallunterscheidungen mit switch	67
1.8.5	Wiederholungen	69
1.8.6	Kontrolle mit break und continue	77
1.9	Benutzerdefinierte und zusammengesetzte Datentypen	79
1.9.1	Aufzählungstypen	79
1.9.2	Arrays: Der C++-Standardtyp vector	81
1.9.3	Zeichenketten: Der C++-Standardtyp string	86
1.9.4	Strukturen	88
1.9.5	Typermittlung mit auto	90
1.9.6	Unions und Bitfelder	91
2	Einfache Ein- und Ausgabe	93
2.1	Standardein- und -ausgabe	93
2.2	Ein- und Ausgabe mit Dateien	96
3	Programmstrukturierung	101
3.1	Funktionen	102
3.1.1	Aufbau und Prototypen	102
3.1.2	Gültigkeitsbereiche und Sichtbarkeit in Funktionen	104
3.1.3	Lokale static-Variable: Funktion mit Gedächtnis	105
3.2	Schnittstellen zum Datentransfer	106
3.2.1	Übergabe per Wert	107
3.2.2	Übergabe per Referenz	111
3.2.3	Gefahren bei der Rückgabe von Referenzen	112
3.2.4	Vorgegebene Parameterwerte und variable Parameterzahl	113
3.2.5	Überladen von Funktionen	114
3.2.6	Funktion main()	115
3.2.7	Beispiel Taschenrechnersimulation	116
3.2.8	Spezifikation von Funktionen	121
3.3	Modulare Programmgestaltung	122
3.3.1	Steuerung der Übersetzung nur mit #include	122
3.3.2	Einbinden vorübersetzter Programmteile	123

3.3.3	Dateiübergreifende Gültigkeit und Sichtbarkeit	124
3.3.4	Übersetzungseinheit, Deklaration, Definition	126
3.3.5	Compilerdirektiven und Makros	128
3.4	Funktions-Templates	134
3.4.1	Spezialisierung von Templates	137
3.4.2	Einbinden von Templates	138
3.5	inline-Funktionen	139
3.6	Namensräume	141
3.7	C++-Header	142
3.7.1	Einbinden von C-Funktionen	144
4	Objektorientierung 1	147
4.1	Abstrakte Datentypen	148
4.2	Klassen und Objekte	149
4.2.1	inline-Elementfunktionen	152
4.3	Initialisierung und Konstruktoren	154
4.3.1	Standardkonstruktor	154
4.3.2	Allgemeine Konstruktoren	155
4.3.3	Kopierkonstruktor	158
4.3.4	Typumwandlungskonstruktor	160
4.4	Beispiel: Rationale Zahlen	162
4.4.1	Aufgabenstellung	162
4.4.2	Entwurf	163
4.4.3	Implementation	166
4.5	const-Objekte und Methoden	170
4.6	Destruktoren	171
4.7	Wie kommt man zu Klassen und Objekten? Ein Beispiel	173
4.7.1	Einige Analyse-Überlegungen	174
4.7.2	Formulierung des Szenarios in C++	177
4.8	Gegenseitige Abhängigkeit von Klassen	180
4.9	Konstruktor und mehr vorgeben oder verbieten	182
4.10	Delegierender Konstruktor	182
5	Intermezzo: Zeiger	185
5.1	Zeiger und Adressen	186
5.2	C-Arrays	189
5.2.1	C-Arrays und sizeof	191
5.2.2	Indexoperator bei C-Arrays	191

5.2.3	Initialisierung von C-Arrays	192
5.2.4	Zeigerarithmetik	192
5.3	C-Zeichenketten	193
5.4	Dynamische Datenobjekte	200
5.4.1	Freigeben dynamischer Objekte	203
5.5	Zeiger und Funktionen	205
5.5.1	Parameterübergabe mit Zeigern	205
5.5.2	Parameter des main-Programms	207
5.5.3	Gefahren bei der Rückgabe von Zeigern	208
5.6	this-Zeiger	209
5.7	Mehrdimensionale C-Arrays	209
5.7.1	Statische mehrdimensionale C-Arrays	209
5.7.2	Dynamisch erzeugte mehrdimensionale Arrays	213
5.7.3	Klasse für dynamisches zweidimensionales Array	215
5.8	Binäre Ein-/Ausgabe	220
5.9	Zeiger auf Funktionen	223
5.10	Komplexe Deklarationen lesen	226
5.11	Standard-Typumwandlungen für Zeiger	229
5.12	Zeiger auf Elementfunktionen und -daten	230
5.12.1	Zeiger auf Elementfunktionen	230
5.12.2	Zeiger auf Elementdaten	231
6	Objektorientierung 2	233
6.1	Eine String-Klasse	233
6.1.1	Optimierung der Klasse MeinString	239
6.1.2	friend-Funktionen	241
6.2	Klassenspezifische Daten und Funktionen	242
6.2.1	Klassenspezifische Konstante	245
6.3	Klassen-Templates	246
6.3.1	Ein Stack-Template	246
6.3.2	Stack mit statisch festgelegter Größe	249
6.4	Template-Metaprogrammierung	251
6.5	Variadic Templates: Templates mit variabler Parameterzahl	253
7	Vererbung	257
7.1	Vererbung und Initialisierung	263
7.2	Zugriffsschutz	264
7.3	Typbeziehung zwischen Ober- und Unterklasse	266

7.4	Code-Wiederverwendung	267
7.5	Überschreiben von Funktionen in abgeleiteten Klassen	268
7.6	Polymorphismus	270
7.6.1	Virtuelle Funktionen	270
7.6.2	Abstrakte Klassen	275
7.6.3	Virtueller Destruktor	280
7.7	Probleme der Modellierung mit Vererbung	282
7.8	Mehrfachvererbung	285
7.8.1	Namenskonflikte	288
7.8.2	Virtuelle Basisklassen	289
7.9	Standard-Typumwandlungsoperatoren	292
7.10	Typinformationen zur Laufzeit	295
7.11	Using-Deklaration für Klassen	296
7.12	Private- und Protected-Vererbung	297
8	Fehlerbehandlung	301
8.1	Ausnahmebehandlung	303
8.1.1	Exception-Spezifikation in Deklarationen	306
8.1.2	Exception-Hierarchie in C++	307
8.1.3	Besondere Fehlerbehandlungsfunktionen	308
8.1.4	Erkennen logischer Fehler	309
8.1.5	Arithmetische Fehler / Division durch 0	311
8.2	Speicherbeschaffung mit new	312
8.3	Exception-Sicherheit	315
9	Überladen von Operatoren	317
9.1	Rationale Zahlen – noch einmal	319
9.1.1	Arithmetische Operatoren	319
9.1.2	Ausgabeoperator <<	322
9.2	Eine Klasse für Vektoren	323
9.2.1	Index-Operator []	326
9.2.2	Zuweisungsoperator =	328
9.2.3	Mathematische Vektoren	331
9.2.4	Multiplikationsoperator	332
9.3	Inkrement-Operator ++	334
9.4	Typumwandlungsoperator	337
9.5	Smart Pointer: Operatoren -> und *	339
9.5.1	Smart Pointer und die C++-Standardbibliothek	344

9.6	Objekt als Funktion	344
9.6.1	Lambda-Funktionen	346
9.7	new und delete überladen	347
9.7.1	Speichermanagement mit malloc und free	350
9.7.2	Unterscheidung zwischen Heap- und Stack-Objekten	352
9.7.3	Fehlende delete-Anweisung entdecken	353
9.7.4	Eigene Speicherverwaltung	355
9.7.5	Empfehlungen im Umgang mit new und delete	358
9.8	Mehrdimensionale Matrizen	359
9.8.1	Zweidimensionale Matrix als Vektor von Vektoren	360
9.8.2	Dreidimensionale Matrix	363
9.9	Zuweisung bei Vererbung	365
10	Dateien und Ströme	375
10.1	Ausgabe	377
10.1.1	Formatierung der Ausgabe	377
10.2	Eingabe	380
10.3	Manipulatoren	383
10.3.1	Eigene Manipulatoren	386
10.4	Fehlerbehandlung	387
10.5	Typumwandlung von Dateiobjekten nach bool	389
10.6	Arbeit mit Dateien	390
10.6.1	Positionierung in Dateien	391
10.6.2	Lesen und Schreiben in derselben Datei	392
10.7	Umleitung auf Strings	393
10.8	Ergänzungen	395
11	Einführung in die Standard Template Library (STL)	397
11.1	Container, Iteratoren, Algorithmen	398
11.2	Iteratoren im Detail	403
11.3	Beispiel verkettete Liste	404
12	Reguläre Ausdrücke	409
12.1	Elemente regulärer Ausdrücke	410
12.1.1	Greedy oder lazy?	412
12.2	Interaktive Auswertung	413
12.3	Auszug des regex-APIs	416
12.4	Anwendungen	418

13	Threads	419
13.1	Die Klasse thread.....	423
13.2	Synchronisation.....	426
13.2.1	Thread-Group.....	428
13.3	Thread-Steuerung: pausieren, fortsetzen, beenden.....	429
13.4	Interrupt.....	434
13.5	Warten auf Ereignisse	436
13.6	Reader/Writer-Problem.....	441
13.6.1	Wenn Threads verhungern.....	446
13.6.2	Reader/Writer-Varianten	447
13.7	Thread-Sicherheit	448
Teil II: Bausteine komplexer Anwendungen		449
14	Grafische Benutzungsschnittstellen	451
14.1	Ereignisgesteuerte Programmierung.....	452
14.2	GUI-Programmierung mit Qt.....	453
14.2.1	Meta-Objektsystem	453
14.2.2	Der Programmablauf	454
14.2.3	Speicher sparen und lokal Daten sichern	455
14.3	Signale, Slots und Widgets	457
14.4	Dialog	465
14.5	Qt oder Boost?.....	468
14.5.1	Threads	469
14.5.2	Verzeichnisbaum durchwandern	470
15	Internet-Anbindung	473
15.1	Protokolle	474
15.2	Adressen.....	474
15.3	Socket	478
15.3.1	Bidirektionale Kommunikation.....	481
15.3.2	UDP-Sockets	483
15.3.3	Atomuhr mit UDP abfragen	485
15.4	HTTP	488
15.4.1	Verbindung mit GET	489
15.4.2	Verbindung mit POST	494
15.5	Mini-Webserver	494

16	Datenbankanbindung	503
16.1	C++-Interface.....	504
16.2	Anwendungsbeispiel.....	508
Teil III: Praktische Methoden und Werkzeuge der Softwareentwicklung		515
17	Abläufe automatisieren mit make	517
17.1	Quellen	518
17.2	Wirkungsweise	519
17.3	Variablen und Muster	521
17.4	Universelles Makefile für einfache Projekte	522
18	Unit-Test	525
18.1	Werkzeuge	526
18.2	Test Driven Development	527
18.3	Boost Unit Test Framework	528
18.3.1	Beispiel: Testgetriebene Entwicklung einer Operatorfunktion	530
18.3.2	Fixture	534
18.3.3	Testprotokoll und Log-Level	535
18.3.4	Prüf-Makros	536
18.3.5	Kommandozeilen-Optionen.....	540
19	Werkzeuge zur Verwaltung von Projekten	541
19.1	Dokumentation und Strukturanalyse mit doxygen.....	541
19.1.1	Strukturanalyse.....	545
19.2	Versionskontrolle	546
19.2.1	Einrichtung des Servers	548
19.2.2	Exemplarische Benutzung	550
19.3	Projektverwaltung	553
19.3.1	Projektmanagement	553
19.3.2	Wiki für Software-Entwicklungsprojekte	553
Teil IV: Das C++-Rezeptbuch: Tipps und Lösungen für typische Aufgaben.....		555
20	Sichere Programmentwicklung.....	557
20.1	Regeln zum Design von Methoden	557
20.2	Defensive Programmierung.....	560
20.2.1	double- und float-Werte richtig vergleichen	561

20.2.2	const verwenden	561
20.2.3	Anweisungen nach for/if/while einklammern	561
20.2.4	int und unsigned/size_t nicht mischen	562
20.2.5	size_t oder auto statt unsigned int verwenden.....	562
20.2.6	Postfix++ mit Präfix++ implementieren	562
20.2.7	Ein Destruktor darf keine Exception werfen	563
20.2.8	Typumwandlungsoperatoren vermeiden	564
20.2.9	explicit-Konstruktoren bevorzugen	564
20.2.10	Leere Standardkonstruktoren vermeiden	564
20.2.11	Kopieren und Zuweisung verbieten	564
20.2.12	Vererbung verbieten	566
20.2.13	Defensiv Objekte löschen	566
20.3	Exception-sichere Beschaffung von Ressourcen.....	567
20.3.1	Sichere Verwendung von shared_ptr	567
20.3.2	shared_ptr für Arrays korrekt verwenden	567
20.3.3	unique_ptr für Arrays korrekt verwenden	568
20.3.4	Exception-sichere Funktion	569
20.3.5	Exception-sicherer Konstruktor	570
20.3.6	Exception-sichere Zuweisung	572
20.4	Aussagefähige Fehlermeldung ohne neuen String erzeugen.....	574
20.5	Empfehlungen zur Thread-Programmierung.....	575
20.5.1	Warten auf die Freigabe von Ressourcen	575
20.5.2	Deadlock-Vermeidung.....	576
20.5.3	notify_all oder notify_one?.....	576
20.5.4	Performance mit Threads verbessern?.....	577
21	Von der UML nach C++	579
21.1	Vererbung	580
21.2	Interface anbieten und nutzen	580
21.3	Assoziation	582
21.3.1	Aggregation.....	585
21.3.2	Komposition	585
22	Performance, Wert- und Referenzsemantik	587
22.1	Performanceproblem Wertsemantik	589
22.1.1	Auslassen der Kopie	589
22.1.2	Temporäre Objekte bei der Zuweisung	590
22.2	Optimierung durch Referenzsemantik für R-Werte.....	591

22.2.1	Bewegender Konstruktor	594
22.2.2	Bewegender Zuweisungsoperator	595
22.3	Ein effizienter binärer Plusoperator	596
22.3.1	Kopien temporärer Objekte eliminieren	597
22.3.2	Verbesserung durch verzögerte Auswertung	597
23	Effektive Programmerzeugung	601
23.1	Automatische Ermittlung von Abhängigkeiten	602
23.1.1	Getrennte Verzeichnisse: src, obj, bin	603
23.2	Makefile für Verzeichnisbäume	605
23.2.1	Rekursive Make-Aufrufe	606
23.2.2	Ein Makefile für alles	608
23.3	Automatische Erzeugung von Makefiles	609
23.3.1	Makefile für rekursive Aufrufe erzeugen	610
23.4	Erzeugen von Bibliotheken	611
23.4.1	Statische Bibliotheksmodule	612
23.4.2	Dynamische Bibliotheksmodule	613
23.5	GNU Autotools	616
23.6	CMake	619
23.7	Code Bloat bei der Instanziierung von Templates vermeiden	619
23.7.1	extern-Template	620
23.7.2	Aufspaltung in Schnittstelle und Implementation	622
24	Algorithmen für verschiedene Aufgaben	623
24.1	Algorithmen mit Strings	624
24.1.1	String splitten	624
24.1.2	String in Zahl umwandeln	625
24.1.3	Zahl in String umwandeln	629
24.1.4	Strings sprachlich richtig sortieren	629
24.1.5	Umwandlung in Klein- bzw. Großschreibung	631
24.1.6	Strings sprachlich richtig vergleichen	633
24.1.7	Von der Groß-/Kleinschreibung unabhängiger Zeichenvergleich	634
24.1.8	Von der Groß-/Kleinschreibung unabhängige Suche	635
24.2	Textverarbeitung	636
24.2.1	Datei durchsuchen	636
24.2.2	Ersetzungen in einer Datei	638
24.2.3	Code-Formatierer	640
24.2.4	Lines of Code (LOC) ermitteln	641

24.2.5	Zeilen, Wörter und Zeichen einer Datei zählen	643
24.2.6	CSV-Datei lesen	643
24.2.7	Kreuzreferenzliste	645
24.3	Operationen auf Folgen	647
24.3.1	Container anzeigen	648
24.3.2	Folge mit gleichen Werten initialisieren	648
24.3.3	Folge mit Werten eines Generators initialisieren	648
24.3.4	Folge mit fortlaufenden Werten initialisieren	649
24.3.5	Summe und Produkt	650
24.3.6	Mittelwert und Standardabweichung	651
24.3.7	Skalarprodukt	651
24.3.8	Folge der Teilsummen oder -produkte	653
24.3.9	Folge der Differenzen	653
24.3.10	Minimum und Maximum	655
24.3.11	Elemente rotieren	656
24.3.12	Elemente verwürfeln	657
24.3.13	Dubletten entfernen	658
24.3.14	Reihenfolge umdrehen	660
24.3.15	Anzahl der Elemente, die einer Bedingung genügen	661
24.3.16	Gilt X für alle, keins oder wenigstens ein Element einer Folge?	662
24.3.17	Permutationen	663
24.3.18	Lexikografischer Vergleich	665
24.4	Sortieren und Verwandtes	666
24.4.1	Partitionieren	666
24.4.2	Sortieren	667
24.4.3	Stabiles Sortieren	667
24.4.4	Partielles Sortieren	669
24.4.5	Das n.-größte oder n.-kleinste Element finden	669
24.4.6	Verschmelzen (merge)	671
24.5	Suchen und Finden	674
24.5.1	Element finden	674
24.5.2	Element einer Menge in der Folge finden	675
24.5.3	Teilfolge finden	677
24.5.4	Bestimmte benachbarte Elemente finden	679
24.5.5	Bestimmte aufeinanderfolgende Werte finden	680
24.5.6	Binäre Suche	681
24.6	Mengenoperationen auf sortierten Strukturen	684

24.6.1	Teilmengenrelation	684
24.6.2	Vereinigung	685
24.6.3	Schnittmenge	686
24.6.4	Differenz	686
24.6.5	Symmetrische Differenz	687
24.7	Heap-Algorithmen	688
24.7.1	pop_heap	689
24.7.2	push_heap	690
24.7.3	make_heap	691
24.7.4	sort_heap	691
24.7.5	is_heap	692
24.8	Vergleich von Containern auch ungleichen Typs	692
24.8.1	Unterschiedliche Elemente finden	692
24.8.2	Prüfung auf gleiche Inhalte	694
24.9	Rechnen mit komplexen Zahlen: Der C++-Standardtyp complex	695
24.10	Schnelle zweidimensionale Matrix	697
24.10.1	Optimierung mathematischer Array-Operationen	701
24.11	Singleton	705
24.11.1	Implementierung mit einem Zeiger	706
24.11.2	Implementierung mit einer Referenz	706
24.11.3	Meyers' Singleton	707
24.12	Vermischtes	710
24.12.1	Erkennung eines Datums	710
24.12.2	Erkennung einer IP-Adresse	712
24.12.3	Erzeugen von Zufallszahlen	712
24.12.4	for_each – Auf jedem Element eine Funktion ausführen	716
24.12.5	Verschiedene Möglichkeiten, Container-Bereiche zu kopieren	717
24.12.6	Vertauschen von Elementen, Bereichen und Containern	719
24.12.7	Elemente transformieren	720
24.12.8	Ersetzen und Varianten	722
24.12.9	Elemente herausfiltern	723
24.12.10	Grenzwerte von Zahltypen	725
24.12.11	Minimum und Maximum	726
25	Ein- und Ausgabe	727
25.1	Datei- und Verzeichnisoperationen	727
25.1.1	Datei oder Verzeichnis löschen	728

25.1.2	Datei oder Verzeichnis umbenennen	729
25.1.3	Verzeichnis anlegen.....	730
25.1.4	Verzeichnis anzeigen	731
25.1.5	Verzeichnisbaum anzeigen.....	732
25.2	Tabelle formatiert ausgeben	734
25.3	Formatierte Daten lesen.....	735
25.3.1	Eingabe benutzerdefinierter Typen	735
25.4	Array als Block lesen oder schreiben.....	737

Teil V: Die C++-Standardbibliothek 739

26 Aufbau und Übersicht 741

26.1	Auslassungen	743
26.2	Beispiele des Buchs und die C++-Standardbibliothek	745

27 Hilfsfunktionen und -klassen 747

27.1	Relationale Operatoren	747
27.2	Unterstützung der Referenzsemantik für R-Werte	748
27.3	Paare	750
27.4	Tupel	752
27.5	Funktionsobjekte	753
27.5.1	Arithmetische, vergleichende und logische Operationen	753
27.5.2	Funktionsobjekte zum Negieren logischer Prädikate	753
27.5.3	Binden von Argumentwerten	754
27.5.4	Funktionen in Objekte umwandeln	756
27.6	Templates für rationale Zahlen	758
27.7	Zeit und Dauer	760
27.8	Hüllklasse für Referenzen	761

28 Container 763

28.1	Gemeinsame Eigenschaften	765
28.1.1	Initialisierungslisten	767
28.1.2	Konstruktion an Ort und Stelle	768
28.1.3	Reversible Container	768
28.2	Sequenzen	769
28.2.1	vector	770
28.2.2	vector<bool>	771
28.2.3	list	772

28.2.4	deque	775
28.2.5	stack	776
28.2.6	queue	777
28.2.7	priority_queue	778
28.2.8	array	780
28.3	Sortierte assoziative Container	782
28.3.1	map	782
28.3.2	multimap	787
28.3.3	set	787
28.3.4	multiset	791
28.4	Hash-Container	791
28.4.1	unordered_map	793
28.4.2	unordered_multimap	798
28.4.3	unordered_set	798
28.4.4	unordered_multiset	801
28.5	bitset	801
29	Iteratoren	805
29.1	Iterator-Kategorien	806
29.1.1	Anwendung von Traits	808
29.2	distance() und advance()	810
29.3	Reverse-Iteratoren	811
29.4	Insert-Iteratoren	812
29.5	Stream-Iteratoren	813
30	Algorithmen	815
30.1	Algorithmen mit Prädikat	816
30.1.1	Algorithmen mit binärem Prädikat	816
30.2	Übersicht	817
31	Nationale Besonderheiten	821
31.1	Sprachumgebungen festlegen und ändern	822
31.1.1	Die locale-Funktionen	823
31.2	Zeichensätze und -codierung	825
31.3	Zeichenklassifizierung und -umwandlung	829
31.4	Kategorien	829
31.4.1	collate	830
31.4.2	ctype	831

31.4.3	numeric	832
31.4.4	monetary	833
31.4.5	time	836
31.4.6	messages	839
31.5	Konstruktion eigener Facetten	839
32	String	841
33	Speichermanagement	849
33.1	Smart Pointer unique_ptr, shared_ptr, weak_ptr	849
33.2	new mit vorgegebenem Speicherort	854
33.3	Hilfsfunktionen	855
34	Optimierte numerische Arrays (valarray)	857
34.1	Konstruktoren	858
34.2	Elementfunktionen	858
34.3	Binäre Valarray-Operatoren	861
34.4	Mathematische Funktionen	863
34.5	slice und slice_array	864
34.6	gslice und gslice_array	866
34.7	mask_array	869
34.8	indirect_array	870
35	C-Header	873
35.1	<cassert>	874
35.2	<cctype>	874
35.3	<cerrno>	875
35.4	<cmath>	875
35.5	<cstdarg>	876
35.6	<cstddef>	877
35.7	<cstdio>	877
35.8	<cstdlib>	877
35.9	<cstring>	879
35.10	<ctime>	881
A	Anhang	883
A.1	Programmierhinweise	883
A.2	C++-Schlüsselwörter	887
A.3	ASCII-Tabelle	887

A.4	Rangfolge der Operatoren	890
A.5	Compilerbefehle	891
A.6	Lösungen zu den Übungsaufgaben	892
A.7	Installation der DVD-Software für Windows.....	937
A.7.1	Installation des Compilers und der Entwicklungsumgebung.....	937
A.7.2	Installation der Boost-Bibliothek	937
A.7.3	Installation von Qt	938
A.7.4	Codeblocks einrichten	938
A.7.5	Integration von Qt in ein Code::Blocks-Projekt.....	940
A.7.6	Bei Verzicht auf die automatische Installation	941
A.8	Installation der DVD-Software für Linux.....	942
A.8.1	Installation des Compilers	942
A.8.2	Installation von Boost	943
A.8.3	Installation von Code::Blocks	944
A.8.4	Code::Blocks einrichten	945
A.8.5	Beispieldateien entpacken	946
A.8.6	Installation von Qt4	946
A.8.7	Integration von Qt in ein Code::Blocks-Projekt.....	947
Glossar		949
Literaturverzeichnis		959
Register		963

Vorwort

Die zweite Auflage dieses Buchs unterscheidet sich im Wesentlichen durch Aktualisierungen im Verlauf der Entwicklung des ISO C++-Standards. Das Buch ist konform zum neuen C++-Standard, ohne den Anspruch auf Vollständigkeit zu erheben – das Dokument [ISOC++] umfasst mehr als 1300 Seiten. Das Buch gibt eine kompetente Einführung in die Sprache und im Teil »Das C++ Rezeptbuch« zahlreiche Tipps und Lösungen für typische Aufgaben, die in der täglichen Praxis anfallen. Es gibt konkrete, sofort umsetzbare Lösungsvorschläge zur defensiven Programmierung, zur exception-sicheren Programmierung, zur Vermeidung von Memory-Problemen, zur Performance-Verbesserung und zur automatisierten Programm- und Bibliothekserzeugung. Zahlreiche Algorithmen für praxisnahe Problemstellungen helfen bei der täglichen Arbeit. Auf größtmögliche Portabilität wird geachtet: Die Beispiele funktionieren unter Linux genau so wie unter Windows. Die problembezogene Orientierung lässt die in die Sprache einführenden Teile kürzer werden. Damit wird das Lernen erleichtert, und die Qualität des Buchs als Nachschlagewerk bleibt erhalten.

Für wen ist dieses Buch geschrieben?

Dieses Buch ist für alle geschrieben, die einen kompakten und gleichzeitig gründlichen Einstieg in die Konzepte und Programmierung mit C++ suchen. Es ist für Anfänger¹ gedacht, die noch keine Programmiererfahrung haben, aber auch für Programmierer, die diese Programmiersprache kennen lernen möchten. Beiden Gruppen und auch C++-Erfahrenen dient das Buch als ausführliches Nachschlagewerk.

Die ersten 11 Kapitel führen in die Sprache ein. Es wird sehr schnell ein Verständnis des objektorientierten Ansatzes entwickelt. Die sofortige praktische Umsetzung des Gelernten steht im Vordergrund. C++ wird als Programmiersprache unabhängig von speziellen Produkten beschrieben; C-Kenntnisse werden nicht vorausgesetzt. Das Buch eignet sich zum Selbststudium und als Begleitbuch zu einer Vorlesung oder zu Kursen. Die vielen Beispiele sind leicht nachzuvollziehen und praxisnah umsetzbar. Klassen und Objekte,

¹ Geschlechtsbezogene Formen meinen hier und im Folgenden stets Männer *und* Frauen.

Templates, STL und Exceptions sind Ihnen bald keine Fremdworte mehr. Es gibt mehr als 85 Übungsaufgaben – mit Musterlösungen im Anhang. Durch das Studium dieser Kapitel werden aus Anfängern bald Fortgeschrittene.

Diesen und anderen Fortgeschrittenen und Profis bietet das Buch eine Einführung in die Themen Thread-Programmierung, Netzwerk-Programmierung mit Sockets einschließlich eines kleinen Webserver, Datenbankbindung, grafische Benutzungsoberflächen und mehr. Dabei wird durch Einsatz der Boost-Library und des Qt-Frameworks größtmögliche Portabilität erreicht.

Softwareentwicklung ist nicht nur Programmierung: Einführend werden anhand von Beispielen unter anderem die Automatisierung der Programmerzeugung mit Make, die Dokumentationserstellung mit Doxygen und die Versionskontrolle mit Subversion behandelt. Das Programmdesign wird durch konkrete Umsetzungen von UML-Mustern nach C++ unterstützt. Das integrierte »C++-Rezeptbuch« mit mehr als 150 praktischen Lösungen, das sehr umfangreiche Register und das detaillierte Inhaltsverzeichnis machen das Buch zu einem praktischen Nachschlagewerk für alle, die sich mit der Softwareentwicklung in C++ beschäftigen.

Übersicht

Schwerpunkt von Teil I ist die Einführung in die Programmiersprache. Die anschließenden Teile gehen darüber hinaus und konzentrieren sich auf die verschiedenen Probleme der täglichen Praxis.

Teil I - Einführung in C++

Das *Kapitel 1* vermittelt zunächst die Grundlagen, wie ein Programm geschrieben und zum Laufen gebracht wird. Es folgen einfache Datentypen und Anweisungen zur Kontrolle des Programmablaufs. Die Einführung der C++-Datentypen `vector` und `string` beendet das Kapitel. *Kapitel 2* beschäftigt sich mit der einfachen Ein- und Ausgabe, auch mit Dateien. Das *Kapitel 3* zeigt Ihnen, wie Sie Funktionen schreiben. Makros, Templates für Funktionen und die modulare Gestaltung von Programmen folgen.

Objektorientierung ist der Schwerpunkt von *Kapitel 4*. Dabei geht es nicht nur um die Konstruktion von Objekten, sondern auch um den Weg von der Problemstellung zu Klassen und Objekten. Zeiger, einfache Arrays (C-Arrays) und Zeichenketten sowie die Erzeugung von Objekten zur Laufzeit sind Inhalt von *Kapitel 5*. Dazu kommen mehrdimensionale C-Arrays und das Schreiben und Lesen von Binärdaten in bzw. aus Dateien. Auf der Basis von Zeigern führt *Kapitel 6* das Thema Objektorientierung fort. Dabei lernen Sie kennen, wie eine String-Klasse funktioniert, und wie Sie Klassen-Templates und Templates mit einer variablen Anzahl von Parametern konstruieren. Das *Kapitel 7* zeigt Ihnen das Mittel objektorientierter Sprachen, um Generalisierungs- und Spezialisierungsbeziehungen auszudrücken: die Vererbung mit ihren Möglichkeiten. Strategien zur Fehlerbehandlung mit Exceptions finden Sie in *Kapitel 8*. Das *Kapitel 9* zeigt, wie Sie Operatorsymbolen wie `+` und `-` eigene Bedeutungen zuweisen können und in welchem Zusammenhang das sinnvoll ist. Sie lernen, »intelligente« Zeiger (Smart Pointer) zu konstruieren und Objekte als Funktionen einzusetzen. *Kapitel 10* beschreibt ausführlich die Ein- und Ausgabemöglichkeiten, die in Kapitel 2 nur einführend gestreift werden, einschließlich der Fehlerbehandlung und der Formatierung der Ausgabe. Eine Einführung

in die Standard Template Library (STL) bietet *Kapitel 11*. Es zeigt, wie die Komponenten (Container, Iteratoren und Algorithmen) zusammenwirken. Die STL und ihre Wirkungsweise bilden die Grundlage eines sehr großen Teils der C++-Standardbibliothek.

Die Kapitel 1 bis 11 sind für ein gutes Verständnis von C++ unverzichtbar. Reguläre Ausdrücke (*Kapitel 12*) und die Programmierung paralleler Abläufe mit Threads (*Kapitel 13*) sind dazu nicht notwendig – hier handelt es sich um Ergänzungen, wie sie von vielen Programmiersprachen angeboten werden.

Teil II - Bausteine komplexer Anwendungen

Ein Programm benötigt eine Möglichkeit, mit der Außenwelt zu kommunizieren. Tastatur und Konsole allein reichen für komplexe Anwendungen in der Regel nicht aus. Mausbedienung und Bildschirmgrafik sind heute Standard bei Desktop-Anwendungen. Das *Kapitel 14* zeigt, wie grafische Benutzungsschnittstellen konstruiert werden. Wie ein Programm die Verbindung mit dem Internet aufnehmen kann, dokumentiert das *Kapitel 15*. Und wohin mit den ganzen Daten, die bei Programmende nicht verloren gehen sollen? In *Kapitel 16* lernen Sie, wie ein Programm an eine Datenbank angebunden wird. Die genannten Themen sind so umfangreich, dass sie selbst Bücher füllen. Dieser Teil bietet Ihnen daher nur einen Einstieg.

Teil III - Praktische Methoden und Werkzeuge der Softwareentwicklung

Die Entwicklung von Programmen besteht nicht nur im Schreiben von Code. Die Compilation eines Projekts mit vielen Programmdateien und Abhängigkeiten kann schnell ein komplexer Vorgang werden. Die Automatisierung dieses Prozesses mit dem Tool *make* ist Thema von *Kapitel 17*. Programme sind nicht auf Anhieb fehlerfrei. Sie müssen getestet werden: *Kapitel 18* stellt ein Werkzeug für den Unit-Test vor und zeigt den praktischen Einsatz. *Kapitel 19* demonstriert ein Werkzeug zur automatischen Dokumentationserstellung, zeigt, wie eine Versionsverwaltung eingerichtet wird, und geht kurz auf die Projektverwaltung ein.

Teil IV - Das C++-Rezeptbuch: Tipps und Lösungen für typische Aufgaben

Sichere Programmentwicklung ist die Überschrift des *Kapitels 20*. Sie finden dort Regeln zum Design von Methoden und mehrere Tipps zur defensiven Programmierung, die die Risiken falscher Algorithmen oder falscher Benutzung vermindern. Auch gibt es Tipps zur exception-sicheren Beschaffung von Speicher und zur Thread-Programmierung. *Kapitel 21* zeigt Rezepte, wie Sie bestimmte UML-Muster in C++-Konstruktionen umwandeln können. *Kapitel 22* erklärt den Unterschied zwischen Wert- und Referenzsemantik und die Auswirkung auf die Geschwindigkeit von C++-Programmen. Es werden Empfehlungen gegeben und am Beispiel konkretisiert, wie die Performanz deutlich verbessert werden kann. *Kapitel 23* erweitert die Grundlagen des Kapitels 17 um praktische Rezepte zur automatischen Ermittlung von Abhängigkeiten zwischen Programmdateien, Makefiles für Verzeichnisbäume, die automatische Erzeugung von Makefiles und statischer und dynamischer Bibliotheken. Algorithmen für viele verschiedene Aufgaben finden Sie in *Kapitel 24*. Wegen der Vielzahl empfiehlt sich ein Blick in das Inhaltsverzeichnis, um einen Überblick zu gewinnen. Der C++-Standard bietet für viele Datei- und Verzeichnisoperationen keine Unterstützung an. *Kapitel 25* enthält fertige Rezepte zum Anlegen,

Löschen und Lesen von Verzeichnissen und mehr auf der Basis der Boost-Library. Ergänzt wird dies durch das formatierte Lesen und Schreiben von Daten und die Abspeicherung binärer Daten als Block.

Teil V - Die C++-Standardbibliothek

In mehreren Kapiteln wird die C++-Standardbibliothek in Kürze beschrieben. Die Inhalte dieses Teils sind: Hilfsfunktionen und -klassen, Container, Iteratoren, Algorithmen, Einstellung nationaler Besonderheiten, String, Speicherverwaltung, Funktionen der Programmiersprache C.

Anhang

Der Anhang enthält unter anderem Empfehlungen zur Programmierung, verschiedene hilfreiche Tabellen und die Lösungen der Übungsaufgaben.

Wo finden Sie was?

Bei der Programmentwicklung wird häufig das Problem auftauchen, etwas nachschlagen zu müssen. Es gibt die folgenden Hilfen:

Erklärungen zu Begriffen sind im *Glossar* ab Seite 949 aufgeführt.

Es gibt ein recht umfangreiches *Stichwortverzeichnis* ab Seite 963 und ein sehr detailliertes *Inhaltsverzeichnis*.

Auf der Webseite <http://www.cppbuch.de/> finden Sie weiteres Material, Hinweise, Errata und nützliche Links.

Zu guter Letzt

Allen Menschen, die dieses Buch durch Hinweise und Anregungen verbessern halfen, sei an dieser Stelle herzlich gedankt. Frau Margarete Metzger und Frau Irene Weihart vom Hanser Verlag danke ich für die sehr gute Zusammenarbeit.

Bremen, im Juni 2011

Ulrich Breymann