

Inhalt

Vorwort	23
Teil I: Einführung in C++	27
1 Es geht los!	29
1.1 Historisches	29
1.2 Objektorientierte Programmierung	30
1.3 Werkzeuge zum Programmieren	32
1.4 Das erste Programm	33
1.4.1 Namenskonventionen	39
1.5 Integrierte Entwicklungsumgebung	39
1.6 Einfache Datentypen und Operatoren	42
1.6.1 Ausdruck	42
1.6.2 Ganze Zahlen	42
1.6.3 Reelle Zahlen	49
1.6.4 Konstanten	53
1.6.5 Zeichen	54
1.6.6 Logischer Datentyp bool	58
1.6.7 Referenzen	59
1.6.8 Regeln zum Bilden von Ausdrücken	60
1.6.9 Standard-Typumwandlungen	61
1.7 Gültigkeitsbereich und Sichtbarkeit	62
1.7.1 Namespace std	64

1.8	Kontrollstrukturen	65
1.8.1	Anweisungen	65
1.8.2	Sequenz (Reihung)	67
1.8.3	Auswahl (Selektion, Verzweigung)	67
1.8.4	Fallunterscheidungen mit switch	72
1.8.5	Wiederholungen.....	75
1.8.6	Kontrolle mit break und continue	82
1.9	Benutzerdefinierte und zusammengesetzte Datentypen	84
1.9.1	Aufzählungstypen	84
1.9.2	Strukturen.....	87
1.9.3	Der C++-Standardtyp vector	88
1.9.4	Zeichenketten: Der C++-Standardtyp string	93
1.9.5	Container und Schleifen	95
1.9.6	Typermittlung mit auto	97
1.9.7	Deklaration einer strukturierten Bindung mit auto.....	98
1.9.8	Unions und Bitfelder	99
1.10	Einfache Ein- und Ausgabe	101
1.10.1	Standardein- und -ausgabe.....	101
1.10.2	Ein- und Ausgabe mit Dateien	104
2	Programmstrukturierung	109
2.1	Funktionen.....	110
2.1.1	Aufbau und Prototypen	110
2.1.2	Gültigkeitsbereiche und Sichtbarkeit in Funktionen	112
2.1.3	Lokale static-Variable: Funktion mit Gedächtnis	113
2.2	Schnittstellen zum Datentransfer	114
2.2.1	Übergabe per Wert	115
2.2.2	Übergabe per Referenz	119
2.2.3	Gefahren bei der Rückgabe von Referenzen.....	120
2.2.4	Vorgegebene Parameterwerte und unterschiedliche Parameterzahl	121
2.2.5	Überladen von Funktionen	122
2.2.6	Funktion main()	123
2.2.7	Beispiel Taschenrechnersimulation	124
2.2.8	Spezifikation von Funktionen	129
2.2.9	Alternative Funktions-Syntax.....	129
2.3	Modulare Programmgestaltung	129
2.3.1	Steuerung der Übersetzung nur mit #include	130

2.3.2	Einbinden vorübersetzter Programmteile	131
2.3.3	Übersetzungseinheit, Deklaration, Definition	132
2.3.4	Dateiübergreifende Gültigkeit und Sichtbarkeit.....	134
2.3.5	Präprozessordirektiven und Makros	136
2.4	Namensräume	144
2.5	inline-Funktionen und -Variable	146
2.5.1	inline-Variablen	147
2.6	constexpr-Funktionen	147
2.7	Rückgabety auto.....	150
2.8	Funktions-Templates	151
2.8.1	Spezialisierung von Templates	153
2.8.2	Einbinden von Templates	154
2.9	C++-Header.....	156
2.9.1	Einbinden von C-Funktionen	158
3	Objektorientierung 1	159
3.1	Abstrakter Datentyp	160
3.2	Klassen und Objekte	161
3.2.1	constexpr-Objekte und Methoden.....	164
3.2.2	inline-Elementfunktionen.....	165
3.3	Initialisierung und Konstruktoren	166
3.3.1	Standardkonstruktor.....	167
3.3.2	Direkte Initialisierung der Attribute	168
3.3.3	Allgemeine Konstruktoren	168
3.3.4	Kopierkonstruktor	171
3.3.5	Typumwandlungskonstruktor	174
3.3.6	Konstruktor und mehr vorgeben oder verbieten.....	176
3.3.7	Einheitliche Initialisierung und Sequenzkonstruktor.....	176
3.3.8	Delegierender Konstruktor	179
3.3.9	constexpr-Konstruktor und -Methoden	180
3.4	Beispiel: Rationale Zahlen	183
3.4.1	Aufgabenstellung	183
3.4.2	Entwurf.....	184
3.4.3	Implementation.....	187
3.5	Destruktoren	192
3.6	Wie kommt man zu Klassen und Objekten? Ein Beispiel.....	194
3.7	Gegenseitige Abhängigkeit von Klassen	199

4	Intermezzo: Zeiger	201
4.1	Zeiger und Adressen.....	202
4.2	C-Arrays.....	205
4.2.1	C-Array, std::size() und sizeof	207
4.2.2	Initialisierung von C-Arrays.....	208
4.2.3	Zeigerarithmetik.....	208
4.2.4	Indexoperator bei C-Arrays.....	209
4.2.5	C-Array mit begin() und end() durchlaufen	209
4.3	C-Zeichenketten	210
4.3.1	Schleifen und C-Strings.....	213
4.4	Dynamische Datenobjekte	217
4.4.1	Freigeben dynamischer Objekte.....	220
4.5	Zeiger und Funktionen.....	222
4.5.1	Parameterübergabe mit Zeigern.....	222
4.5.2	Array als Funktionsparameter.....	224
4.5.3	const und Zeiger-Parameter	225
4.5.4	Parameter des main-Programms	225
4.5.5	Gefahren bei der Rückgabe von Zeigern.....	226
4.6	this-Zeiger	227
4.7	Mehrdimensionale C-Arrays.....	229
4.7.1	Statische mehrdimensionale C-Arrays	229
4.7.2	Mehrdimensionales Array als Funktionsparameter.....	230
4.7.3	Dynamisch erzeugte mehrdimensionale Arrays.....	233
4.7.4	Klasse für dynamisches zweidimensionales Array	235
4.8	Binäre Ein-/Ausgabe	241
4.9	Zeiger auf Funktionen.....	244
4.10	Typumwandlungen für Zeiger.....	248
4.11	Zeiger auf Elementfunktionen und -daten	249
4.11.1	Zeiger auf Elementfunktionen.....	249
4.11.2	Zeiger auf Elementdaten	250
4.12	Komplexe Deklarationen lesen	250
4.12.1	Lesbarkeit mit typedef und using verbessern	251
4.13	Alternative zu rohen Zeigern, new und delete	253
5	Objektorientierung 2	255
5.1	Eine String-Klasse	255
5.1.1	friend-Funktionen	261

5.2	String-Ansicht.....	262
5.3	Klassenspezifische Daten und Funktionen	265
5.3.1	Klassenspezifische Konstante.....	269
5.4	Klassen-Templates	271
5.4.1	Ein Stack-Template	271
5.4.2	Stack mit statisch festgelegter Größe.....	274
5.5	Typbestimmung mit decltype und declval.....	275
6	Vererbung	279
6.1	Vererbung und Initialisierung	285
6.2	Zugriffsschutz	286
6.3	Typbeziehung zwischen Ober- und Unterklasse	288
6.4	Code-Wiederverwendung	289
6.4.1	Konstruktor erben.....	290
6.5	Überschreiben von Funktionen in abgeleiteten Klassen.....	292
6.5.1	Virtuelle Funktionen.....	294
6.5.2	Abstrakte Klassen	299
6.5.3	Virtueller Destruktor.....	304
6.5.4	Private virtuelle Funktionen.....	307
6.6	Probleme der Modellierung mit Vererbung.....	309
6.7	Mehrfachvererbung.....	312
6.7.1	Namenskonflikte	314
6.7.2	Virtuelle Basisklassen	315
6.8	Standard-Typumwandlungsoperatoren	319
6.9	Typinformationen zur Laufzeit.....	322
6.10	Using-Deklaration für protected-Funktionen	323
6.11	Private- und Protected-Vererbung.....	324
7	Fehlerbehandlung.....	329
7.1	Ausnahmebehandlung	331
7.1.1	Exception-Spezifikation in Deklarationen	334
7.1.2	Exception-Hierarchie in C++	335
7.1.3	Besondere Fehlerbehandlungsfunktionen.....	337
7.1.4	Erkennen logischer Fehler	338
7.1.5	Arithmetische Fehler / Division durch 0	340
7.2	Speicherbeschaffung mit new	341
7.3	Exception-Sicherheit	342

8	Überladen von Operatoren	345
8.1	Rationale Zahlen – noch einmal	347
8.1.1	Arithmetische Operatoren	347
8.1.2	Ausgabeoperator <<	349
8.2	Eine Klasse für Vektoren	351
8.2.1	Index-Operator []	354
8.2.2	Zuweisungsoperator =	356
8.2.3	Mathematische Vektoren	359
8.2.4	Multiplikationsoperator	360
8.3	Inkrement-Operator ++	362
8.4	Typumwandlungsoperator	366
8.5	Smart Pointer: Operatoren -> und *	367
8.5.1	Smart Pointer und die C++-Standardbibliothek	372
8.6	Objekt als Funktion	373
8.7	new und delete überladen	375
8.7.1	Unterscheidung zwischen Heap- und Stack-Objekten	378
8.7.2	Fehlende delete-Anweisung entdecken	380
8.7.3	Eigene Speicherverwaltung für einen bestimmten Typ	381
8.7.4	Empfehlungen im Umgang mit new und delete	386
8.8	Operatoren für Literale	386
8.8.1	Stringliterals	387
8.8.2	Benutzerdefinierte Literale	388
8.9	Mehrdimensionale Matrizen	390
8.9.1	Zweidimensionale Matrix als Vektor von Vektoren	391
8.9.2	Dreidimensionale Matrix	394
8.10	Zuweisung und Vergleich bei Vererbung	396
9	Dateien und Ströme	405
9.1	Ausgabe	407
9.1.1	Formatierung der Ausgabe	407
9.2	Eingabe	410
9.3	Manipulatoren	413
9.3.1	Eigene Manipulatoren	418
9.4	Fehlerbehandlung	420
9.5	Typumwandlung von Dateiobjekten nach bool	421
9.6	Arbeit mit Dateien	422
9.6.1	Positionierung in Dateien	423

9.6.2	Lesen und Schreiben in derselben Datei	424
9.7	Umleitung auf Strings	425
9.8	Tabelle formatiert ausgeben	427
9.9	Formatierte Daten lesen	428
9.9.1	Eingabe benutzerdefinierter Typen	428
9.10	Blockweise lesen und schreiben	430
9.10.1	vector-Objekt binär lesen und schreiben	430
9.10.2	array-Objekt binär lesen und schreiben	431
9.10.3	Matrix binär lesen und schreiben	432
9.11	Ergänzungen	434
10	Die Standard Template Library (STL)	435
10.1	Container, Iteratoren, Algorithmen	436
10.2	Iteratoren im Detail	441
10.3	Beispiel verkettete Liste	442
Teil II: Fortgeschrittene Themen		447
11	Performance, Wert- und Referenzsemantik	449
11.1	Performanceproblem Wertsemantik	451
11.1.1	Auslassen der Kopie	451
11.1.2	Temporäre Objekte bei der Zuweisung	452
11.2	Referenzsemantik für R-Werte	453
11.3	Optimierung durch Referenzsemantik für R-Werte	455
11.3.1	Bewegender Konstruktor	458
11.3.2	Bewegender Zuweisungsoperator	458
11.4	Die move()-Funktion	459
11.4.1	Regel zur Template-Auswertung von &&-Parametern	461
11.5	Ein effizienter binärer Plusoperator	462
11.5.1	Return Value Optimization (RVO)	463
11.5.2	Kopien temporärer Objekte eliminieren	463
11.5.3	Verbesserung durch verzögerte Auswertung	464
11.5.4	Weitere Optimierungsmöglichkeiten	466
11.6	Rule of three/five/zero	467
11.6.1	Rule of three	467
11.6.2	Rule of five	467
11.6.3	Rule of zero	468

12	Lambda-Funktionen	469
12.1	Eigenschaften	470
12.1.1	Äquivalenz zum Funktionszeiger	471
12.1.2	Lambda-Funktion und Klasse	472
12.2	Generische Lambda-Funktionen	473
12.3	Parametererfassung mit <code>[]</code>	475
13	Template-Metaprogrammierung	477
13.1	Grundlagen	478
13.2	Variadic Templates: Templates mit variabler Parameterzahl	480
13.2.1	Ablauf der Auswertung durch den Compiler	481
13.2.2	Anzahl der Parameter	482
13.2.3	Parameterexpansion	482
13.3	Fold-Expressions	484
13.3.1	Weitere Varianten	485
13.3.2	Fold-Expression mit Kommaoperator	486
13.4	Klassen-Template mit variabler Stelligkeit	488
14	Reguläre Ausdrücke	489
14.1	Elemente regulärer Ausdrücke	490
14.1.1	Greedy oder lazy?	492
14.2	Interaktive Auswertung	494
14.3	Auszug des regex-API	497
14.4	Verarbeitung von <code>\n</code>	498
14.5	Anwendungen	500
15	Threads	501
15.1	Zeit und Dauer	502
15.2	Threads	503
15.3	Die Klasse <code>thread</code>	507
15.3.1	<code>Thread-Group</code>	509
15.4	Synchronisation kritischer Abschnitte	510
15.5	Thread-Steuerung: Pausieren, Fortsetzen, Beenden	513
15.5.1	Data Race	518
15.6	Warten auf Ereignisse	518
15.7	Reader/Writer-Problem	524
15.7.1	Wenn Threads verhungern	528
15.7.2	Reader/Writer-Varianten	529

15.8	Atomare Veränderung von Variablen	529
15.9	Asynchrone verteilte Bearbeitung einer Aufgabe	532
15.10	Thread-Sicherheit	534
16	Grafische Benutzungsschnittstellen	535
16.1	Ereignisgesteuerte Programmierung	536
16.2	GUI-Programmierung mit Qt	537
16.2.1	Installation und Einsatz	537
16.2.2	Meta-Objektsystem	538
16.2.3	Der Programmablauf	539
16.2.4	Ereignis abfragen	540
16.3	Signale, Slots und Widgets	541
16.4	Dialog	550
16.5	Qt oder Standard-C++?	553
16.5.1	Threads	554
16.5.2	Verzeichnisbaum durchwandern	555
17	Internet-Anbindung	557
17.1	Protokolle	558
17.2	Adressen	558
17.3	Socket	562
17.3.1	Bidirektionale Kommunikation	565
17.3.2	UDP-Sockets	567
17.3.3	Atomuhr mit UDP abfragen	569
17.4	HTTP	571
17.4.1	Verbindung mit GET	573
17.4.2	Verbindung mit POST	577
17.5	Mini-Webserver	578
18	Datenbankanbindung	587
18.1	C++-Interface	588
18.2	Anwendungsbeispiel	592
Teil III:	Ausgewählte Methoden und Werkzeuge der Softwareentwicklung	599
19	Effiziente Programmerzeugung mit make	601
19.1	Wirkungsweise	603

19.2	Variablen und Muster	605
19.3	Universelles Makefile für einfache Projekte	606
19.4	Automatische Ermittlung von Abhängigkeiten	608
19.4.1	Getrennte Verzeichnisse: src, obj, bin	609
19.5	Makefile für Verzeichnisbäume	611
19.5.1	Rekursive Make-Aufrufe	612
19.5.2	Ein Makefile für alles	614
19.6	Automatische Erzeugung von Makefiles	615
19.6.1	Makefile für rekursive Aufrufe erzeugen	616
19.7	Erzeugen von Bibliotheken	617
19.7.1	Statische Bibliotheksmodule	618
19.7.2	Dynamische Bibliotheksmodule	619
19.8	Code Bloat bei der Instanziierung von Templates vermeiden	622
19.8.1	extern-Templates	623
19.9	CMake	625
19.10	GNU Autotools	626
20	Unit-Test	627
20.1	Werkzeuge	628
20.2	Test Driven Development	629
20.3	Boost Unit Test Framework	630
20.3.1	Beispiel: Testgetriebene Entwicklung einer Operatorfunktion	632
20.3.2	Fixture	636
20.3.3	Testprotokoll und Log-Level	637
20.3.4	Prüf-Makros	638
20.3.5	Kommandozeilen-Optionen	642
Teil IV: Das C++-Rezeptbuch:		
Tipps und Lösungen für typische Aufgaben		643
21	Sichere Programmentwicklung	645
21.1	Regeln zum Design von Methoden	645
21.2	Defensive Programmierung	647
21.2.1	double- und float-Werte richtig vergleichen	648
21.2.2	const und constexpr verwenden	649
21.2.3	Anweisungen nach for/if/while einklammern	649
21.2.4	int und unsigned/size_t nicht mischen	649

21.2.5	size_t oder auto statt unsigned int verwenden.....	650
21.2.6	Postfix++ mit Präfix++ implementieren	650
21.2.7	Ein Destruktor darf keine Exception werfen	651
21.2.8	explicit-Typumwandlungsoperator bevorzugen	651
21.2.9	explicit-Konstruktor für eine Typumwandlung bevorzugen	651
21.2.10	Leere Standardkonstruktoren vermeiden	651
21.2.11	Mit override Schreibfehler reduzieren.....	651
21.2.12	Kopieren und Zuweisung verbieten	651
21.2.13	Vererbung verbieten	652
21.2.14	Überschreiben einer virtuellen Methode verhindern	653
21.2.15	»Rule of zero« beachten	653
21.2.16	One Definition Rule.....	653
21.2.17	Defensiv Objekte löschen	653
21.2.18	Speicherbeschaffung und -freigabe kapseln.....	654
21.2.19	Programmierrichtlinien einhalten	654
21.3	Exception-sichere Beschaffung von Ressourcen.....	654
21.3.1	Sichere Verwendung von unique_ptr und shared_ptr.....	654
21.3.2	So vermeiden Sie new und delete!.....	655
21.3.3	shared_ptr für Arrays korrekt verwenden	656
21.3.4	unique_ptr für Arrays korrekt verwenden	657
21.3.5	Exception-sichere Funktion	658
21.3.6	Exception-sicherer Konstruktor	658
21.3.7	Exception-sichere Zuweisung	659
21.4	Empfehlungen zur Thread-Programmierung.....	660
21.4.1	Warten auf die Freigabe von Ressourcen	660
21.4.2	Deadlock-Vermeidung.....	661
21.4.3	notify_all oder notify_one?.....	662
21.4.4	Performance mit Threads verbessern?.....	662
22	Von der UML nach C++	663
22.1	Vererbung.....	664
22.2	Interface anbieten und nutzen	664
22.3	Assoziation	666
22.3.1	Aggregation.....	669
22.3.2	Komposition	670
23	Algorithmen für verschiedene Aufgaben.....	671
23.1	Algorithmen mit Strings	672

23.1.1	String splitten	672
23.1.2	String in Zahl umwandeln	673
23.1.3	Zahl in String umwandeln	675
23.1.4	Strings sprachlich richtig sortieren	676
23.1.5	Umwandlung in Klein- bzw. Großschreibung	678
23.1.6	Strings sprachlich richtig vergleichen	680
23.1.7	Von der Groß-/Kleinschreibung unabhängiger Zeichenvergleich	681
23.1.8	Von der Groß-/Kleinschreibung unabhängige Suche	682
23.2	Textverarbeitung	683
23.2.1	Datei durchsuchen	683
23.2.2	Ersetzungen in einer Datei	685
23.2.3	Lines of Code (LOC) ermitteln	687
23.2.4	Zeilen, Wörter und Zeichen einer Datei zählen	688
23.2.5	CSV-Datei lesen	688
23.2.6	Kreuzreferenzliste	690
23.3	Operationen auf Folgen	692
23.3.1	Container anzeigen	693
23.3.2	Folge mit gleichen Werten initialisieren	693
23.3.3	Folge mit Werten eines Generators initialisieren	694
23.3.4	Folge mit fortlaufenden Werten initialisieren	694
23.3.5	Summe und Produkt	695
23.3.6	Mittelwert und Standardabweichung	696
23.3.7	Skalarprodukt	696
23.3.8	Folge der Teilsummen oder -produkte	697
23.3.9	Folge der Differenzen	698
23.3.10	Kleinstes und größtes Element finden	699
23.3.11	Elemente rotieren	701
23.3.12	Elemente verwürfeln	702
23.3.13	Dubletten entfernen	702
23.3.14	Reihenfolge umdrehen	704
23.3.15	Stichprobe	705
23.3.16	Anzahl der Elemente, die einer Bedingung genügen	706
23.3.17	Gilt ein Prädikat für alle, keins oder wenigstens ein Element einer Folge?	707
23.3.18	Permutationen	708
23.3.19	Lexikografischer Vergleich	711
23.4	Sortieren und Verwandtes	712
23.4.1	Partitionieren	712

23.4.2	Sortieren.....	714
23.4.3	Stabiles Sortieren	715
23.4.4	Partielles Sortieren	716
23.4.5	Das n.-größte oder n.-kleinste Element finden	717
23.4.6	Verschmelzen (merge)	718
23.5	Suchen und Finden	721
23.5.1	Element finden	721
23.5.2	Element einer Menge in der Folge finden	722
23.5.3	Teilfolge finden.....	723
23.5.4	Teilfolge mit speziellem Algorithmus finden.....	725
23.5.5	Bestimmte benachbarte Elemente finden	726
23.5.6	Bestimmte aufeinanderfolgende Werte finden	727
23.5.7	Binäre Suche.....	728
23.6	Mengenoperationen auf sortierten Strukturen	731
23.6.1	Teilmengenrelation	731
23.6.2	Vereinigung	732
23.6.3	Schnittmenge	733
23.6.4	Differenz	733
23.6.5	Symmetrische Differenz.....	734
23.7	Heap-Algorithmen	735
23.7.1	pop_heap	736
23.7.2	push_heap.....	737
23.7.3	make_heap	737
23.7.4	sort_heap	738
23.7.5	is_heap	738
23.8	Vergleich von Containern auch ungleichen Typs.....	739
23.8.1	Unterschiedliche Elemente finden	739
23.8.2	Prüfung auf gleiche Inhalte	741
23.9	Rechnen mit komplexen Zahlen: Der C++-Standardtyp complex.....	742
23.10	Schnelle zweidimensionale Matrix	744
23.10.1	Optimierung mathematischer Array-Operationen	747
23.11	Vermischtes	750
23.11.1	Erkennung eines Datums.....	750
23.11.2	Erkennung einer IPv4-Adresse	752
23.11.3	Erzeugen von Zufallszahlen	753
23.11.4	for_each – Auf jedem Element eine Funktion ausführen	758
23.11.5	Verschiedene Möglichkeiten, Container-Bereiche zu kopieren.....	758

23.11.6	Vertauschen von Elementen, Bereichen und Containern	761
23.11.7	Elemente transformieren	761
23.11.8	Ersetzen und Varianten	763
23.11.9	Elemente herausfiltern	764
23.11.10	Grenzwerte von Zahltypen	766
23.11.11	Minimum und Maximum	767
23.11.12	Wert begrenzen.....	768
23.11.13	ggT und kgV.....	769
23.12	Parallelisierbare Algorithmen	770
24	Datei- und Verzeichnisoperationen	771
24.1	Übersicht	772
24.2	Pfadoperationen.....	773
24.3	Datei oder Verzeichnis löschen.....	774
24.3.1	Möglicherweise gefülltes Verzeichnis löschen	775
24.4	Datei oder Verzeichnis kopieren	776
24.5	Datei oder Verzeichnis umbenennen	777
24.6	Verzeichnis anlegen	777
24.7	Verzeichnis anzeigen	778
24.8	Verzeichnisbaum anzeigen	779
Teil V:	Die C++-Standardbibliothek	781
25	Aufbau und Übersicht	783
25.1	Auslassungen	785
25.2	Beispiele des Buchs und die C++-Standardbibliothek	787
26	Hilfsfunktionen und -klassen	789
26.1	Relationale Operatoren.....	789
26.2	Unterstützung der Referenzsemantik für R-Werte.....	790
26.2.1	move()	790
26.2.2	forward()	791
26.3	Paare.....	792
26.4	Tupel.....	794
26.5	bitset.....	796
26.6	Indexfolgen	799
26.7	variant statt union	800
26.8	Funktionsobjekte.....	801

26.8.1	Arithmetische, vergleichende und logische Operationen	801
26.8.2	Binden von Argumentwerten.....	802
26.8.3	Funktionen in Objekte umwandeln.....	804
26.9	Templates für <i>rationale</i> Zahlen.....	806
26.10	Hüllklasse für Referenzen.....	808
26.11	Optionale Objekte	808
26.12	Type Traits	810
26.12.1	Wie funktionieren Type Traits? – ein Beispiel.....	811
26.12.2	Abfrage von Eigenschaften.....	813
26.12.3	Abfrage numerischer Eigenschaften.....	815
26.12.4	Typbeziehungen	815
26.12.5	Typumwandlungen	816
26.13	Auswahl weiterer Traits	816
26.13.1	decay.....	816
26.13.2	enable_if	816
26.13.3	conditional	817
26.13.4	default_order	817
27	Container	819
27.1	Gemeinsame Eigenschaften.....	821
27.1.1	Initialisierungslisten	823
27.1.2	Konstruktion an Ort und Stelle.....	823
27.1.3	Reversible Container.....	824
27.2	Sequenzen	825
27.2.1	vector.....	826
27.2.2	vector<bool>	827
27.2.3	list.....	829
27.2.4	deque	831
27.2.5	stack	833
27.2.6	queue	834
27.2.7	priority_queue	836
27.2.8	array	838
27.3	Assoziative Container	840
27.4	Sortierte assoziative Container.....	842
27.4.1	map und multimap	843
27.4.2	set und multiset	847
27.5	Hash-Container.....	849

27.5.1	unordered_map und unordered_multimap	853
27.5.2	unordered_set und unordered_multiset	855
28	Iteratoren	857
28.1	Iterator-Kategorien	858
28.1.1	Anwendung von Traits	859
28.2	Abstand und Bewegen	862
28.3	Zugriff auf Anfang und Ende	863
28.3.1	Reverse-Iteratoren	864
28.4	Insert-Iteratoren	865
28.5	Stream-Iteratoren	866
29	Algorithmen	869
29.1	Algorithmen mit Prädikat	870
29.2	Übersicht	871
30	Nationale Besonderheiten	875
30.1	Sprachumgebung festlegen und ändern	876
30.1.1	Die locale-Funktionen	877
30.2	Zeichensätze und -codierung	879
30.3	Zeichenklassifizierung und -umwandlung	883
30.4	Kategorien	884
30.4.1	collate	884
30.4.2	ctype	885
30.4.3	numeric	887
30.4.4	monetary	888
30.4.5	time	891
30.4.6	messages	893
30.5	Konstruktion eigener Facetten	895
31	String	897
31.1	string_view für String-Literale	907
32	Speichermanagement	909
32.1	unique_ptr	909
32.1.1	make_unique	911
32.2	shared_ptr	912
32.2.1	make_shared	913
32.2.2	Typumwandlung in einen Oberklassentyp	913

32.3	weak_ptr	914
32.4	new mit Speicherortangabe	915
33	Numerische Arrays (valarray)	917
33.1	Konstruktoren	918
33.2	Elementfunktionen	918
33.3	Binäre Valarray-Operatoren	921
33.4	Mathematische Funktionen	923
33.5	slice und slice_array	924
33.6	gslice und gslice_array	927
33.7	mask_array	930
33.8	indirect_array	931
34	Ausgewählte C-Header	933
34.1	<cassert>	933
34.2	<cctype>	934
34.3	<cmath>	934
34.4	<cstdlib>	936
34.5	<stdlib>	936
34.6	<ctime>	937
34.7	<cstring>	939
A	Anhang	941
A.1	ASCII-Tabelle	941
A.2	C++-Schlüsselwörter	943
A.3	Compilerbefehle	944
A.4	Rangfolge der Operatoren	945
A.5	C++-Attribute für den Compiler	947
A.6	Lösungen zu den Übungsaufgaben	947
A.7	Installation der Software für Windows	957
A.7.1	Installation des Compilers und der Entwicklungsumgebung	957
A.7.2	Integrierte Entwicklungsumgebung einrichten	958
A.7.3	De-Installation	958
A.8	Installation der Software für Linux	959
A.8.1	Installation des Compilers	959
A.8.2	Installation von Boost	960
A.8.3	Installation und Einrichtung von Code::Blocks	960
A.8.4	Beispieldateien entpacken	961

A.9	Installationshinweise für OS X	961
A.9.1	Installation von Boost	962
A.9.2	Beispieldateien entpacken	962
	Glossar	963
	Literaturverzeichnis	973
	Register.....	977