

HANSER



Inhaltsverzeichnis

zu

„Der C++-Programmierer“ (3. Auflage)

von Ulrich Breymann

ISBN (Buch): 978-3-446-43894-1

ISBN (E-Book): 978-3-446-43953-5

Weitere Informationen und Bestellungen unter
<http://www.hanser-fachbuch.de/978-3-446-43894-1>

sowie im Buchhandel

© Carl Hanser Verlag München

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 Entwicklungsumgebung	37
1.6 Einfache Datentypen und Operatoren	39
1.6.1 Ausdruck	39
1.6.2 Ganze Zahlen	40
1.6.3 Reelle Zahlen	45
1.6.4 Konstante	49
1.6.5 Zeichen	49
1.6.6 Logischer Datentyp bool	53
1.6.7 Referenzen	54
1.6.8 Regeln zum Bilden von Ausdrücken	55
1.6.9 Standard-Typumwandlungen	56
1.7 Gültigkeitsbereich und Sichtbarkeit	57
1.7.1 Namespace std.....	59

1.8	Kontrollstrukturen	60
1.8.1	Anweisungen	60
1.8.2	Sequenz (Reihung)	61
1.8.3	Auswahl (Selektion, Verzweigung)	62
1.8.4	Fallunterscheidungen mit switch	66
1.8.5	Wiederholungen.....	68
1.8.6	Kontrolle mit break und continue	75
1.9	Benutzerdefinierte und zusammengesetzte Datentypen	78
1.9.1	Aufzählungstypen	78
1.9.2	Strukturen.....	80
1.9.3	Der C++-Standardtyp vector	82
1.9.4	Zeichenketten: Der C++-Standardtyp string	86
1.9.5	Container und Schleifen	89
1.9.6	Typermittlung mit auto	90
1.9.7	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 unterschiedliche Parameterzahl	113
3.2.5	Überladen von Funktionen	114
3.2.6	Funktion main()	116
3.2.7	Beispiel Taschenrechnersimulation	116
3.2.8	Spezifikation von Funktionen.....	121
3.2.9	Alternative Funktions-Syntax.....	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	135
3.4.1	Spezialisierung von Templates	137
3.4.2	Einbinden von Templates	138
3.5	inline-Funktionen.....	140
3.6	constexpr-Funktionen.....	141
3.7	Namensräume	143
3.8	C++-Header	145
4	Objektorientierung 1	147
4.1	Abstrakte Datentypen	148
4.2	Klassen und Objekte	149
4.2.1	inline-Elementfunktionen.....	153
4.3	Initialisierung und Konstruktoren	154
4.3.1	Standardkonstruktor.....	154
4.3.2	Direkte Initialisierung der Attribute	155
4.3.3	Allgemeine Konstruktoren	155
4.3.4	Kopierkonstruktor.....	158
4.3.5	Typumwandlungskonstruktor	161
4.3.6	Konstruktor und mehr vorgeben oder verbieten.....	162
4.3.7	Einheitliche Initialisierung und Sequenzkonstruktor.....	163
4.3.8	Delegierender Konstruktor	165
4.4	Beispiel: Rationale Zahlen	166
4.4.1	Aufgabenstellung	166
4.4.2	Entwurf.....	167
4.4.3	Implementation.....	170
4.5	const-Objekte und Methoden.....	174
4.6	Destruktoren	176
4.7	Wie kommt man zu Klassen und Objekten? Ein Beispiel.....	178
4.8	Gegenseitige Abhängigkeit von Klassen	182
5	Intermezzo: Zeiger	185
5.1	Zeiger und Adressen	186
5.2	C-Arrays.....	189
5.2.1	Initialisierung von C-Arrays.....	191
5.2.2	Zeigerarithmetik.....	191

5.2.3	C-Array und sizeof.....	192
5.2.4	Indexoperator bei C-Arrays.....	193
5.2.5	C-Array mit begin() und end() durchlaufen	193
5.3	C-Zeichenketten.....	194
5.4	Dynamische Datenobjekte	201
5.4.1	Freigeben dynamischer Objekte.....	203
5.5	Zeiger und Funktionen.....	206
5.5.1	Parameterübergabe mit Zeigern.....	206
5.5.2	Parameter des main-Programms	208
5.5.3	Gefahren bei der Rückgabe von Zeigern.....	209
5.6	this-Zeiger	210
5.7	Mehrdimensionale C-Arrays.....	210
5.7.1	Statische mehrdimensionale C-Arrays	210
5.7.2	Dynamisch erzeugte mehrdimensionale Arrays.....	214
5.7.3	Klasse für dynamisches zweidimensionales Array	216
5.8	Binäre Ein-/Ausgabe	221
5.9	Zeiger auf Funktionen.....	224
5.10	Standard-Typumwandlungen für Zeiger.....	228
5.11	Zeiger auf Elementfunktionen und -daten	229
5.11.1	Zeiger auf Elementfunktionen.....	229
5.11.2	Zeiger auf Elementdaten	230
5.12	Komplexe Deklarationen lesen	230
5.12.1	Lesbarkeit mit typedef und using verbessern	231
6	Objektorientierung 2	233
6.1	Eine String-Klasse	233
6.1.1	Optimierung der Klasse MeinString	238
6.1.2	friend-Funktionen	242
6.2	Klassenspezifische Daten und Funktionen	243
6.2.1	Klassenspezifische Konstante.....	247
6.3	Klassen-Templates	248
6.3.1	Ein Stack-Template	248
6.3.2	Stack mit statisch festgelegter Größe.....	251
6.4	Template-Metaprogrammierung	253
6.5	Variadic Templates: Templates mit variabler Parameterzahl	255
6.6	Typbestimmung mit decltype	259
6.6.1	declval	260

7	Vererbung	261
7.1	Vererbung und Initialisierung	267
7.2	Zugriffsschutz	268
7.3	Typbeziehung zwischen Ober- und Unterklasse	270
7.4	Code-Wiederverwendung	271
7.4.1	Konstruktor erben	272
7.5	Überschreiben von Funktionen in abgeleiteten Klassen	274
7.5.1	Virtuelle Funktionen	275
7.5.2	Abstrakte Klassen	280
7.5.3	Virtueller Destruktor	285
7.6	Probleme der Modellierung mit Vererbung	288
7.7	Mehrfachvererbung	291
7.7.1	Namenskonflikte	294
7.7.2	Virtuelle Basisklassen	295
7.8	Standard-Typumwandlungsoperatoren	298
7.9	Typinformationen zur Laufzeit	301
7.10	Using-Deklaration für protected-Funktionen	302
7.11	Private- und Protected-Vererbung	303
8	Fehlerbehandlung	307
8.1	Ausnahmebehandlung	309
8.1.1	Exception-Spezifikation in Deklarationen	313
8.1.2	Exception-Hierarchie in C++	313
8.1.3	Besondere Fehlerbehandlungsfunktionen	314
8.1.4	Erkennen logischer Fehler	316
8.1.5	Arithmetische Fehler / Division durch 0	318
8.2	Speicherbeschaffung mit new	319
8.3	Exception-Sicherheit	321
9	Überladen von Operatoren	323
9.1	Rationale Zahlen – noch einmal	325
9.1.1	Arithmetische Operatoren	325
9.1.2	Ausgabeoperator <<	328
9.2	Eine Klasse für Vektoren	329
9.2.1	Index-Operator []	332
9.2.2	Zuweisungsoperator =	335
9.2.3	Mathematische Vektoren	337
9.2.4	Multiplikationsoperator	339

9.3	Inkrement-Operator ++	340
9.4	Typumwandlungsoperator	344
9.5	Smart Pointer: Operatoren -> und *	346
9.5.1	Smart Pointer und die C++-Standardbibliothek	350
9.6	Objekt als Funktion	351
9.6.1	Lambda-Funktionen	352
9.7	new und delete überladen	354
9.7.1	Speichermanagement mit malloc und free	357
9.7.2	Unterscheidung zwischen Heap- und Stack-Objekten	358
9.7.3	Fehlende delete-Anweisung entdecken	360
9.7.4	Eigene Speicherverwaltung für einen bestimmten Typ	361
9.7.5	Empfehlungen im Umgang mit new und delete	365
9.8	Mehrdimensionale Matrizen	366
9.8.1	Zweidimensionale Matrix als Vektor von Vektoren	367
9.8.2	Dreidimensionale Matrix	369
9.9	Zuweisung bei Vererbung	372
10	Dateien und Ströme	381
10.1	Ausgabe	383
10.1.1	Formatierung der Ausgabe	383
10.2	Eingabe	386
10.3	Manipulatoren	389
10.3.1	Eigene Manipulatoren	393
10.4	Fehlerbehandlung	394
10.5	Typumwandlung von Dateiobjekten nach bool	396
10.6	Arbeit mit Dateien	397
10.6.1	Positionierung in Dateien	398
10.6.2	Lesen und Schreiben in derselben Datei	399
10.7	Umleitung auf Strings	400
10.8	Ergänzungen	402
11	Einführung in die Standard Template Library (STL)	403
11.1	Container, Iteratoren, Algorithmen	404
11.2	Iteratoren im Detail	409
11.3	Beispiel verkettete Liste	410
12	Reguläre Ausdrücke	415
12.1	Elemente regulärer Ausdrücke	416

12.1.1 Greedy oder lazy?	418
12.2 Interaktive Auswertung	419
12.3 Auszug des regex-APIs	422
12.4 Anwendungen	424
13 Threads	425
13.1 Zeit und Dauer	429
13.2 Die Klasse thread	430
13.2.1 Thread-Group	432
13.3 Synchronisation	434
13.4 Thread-Steuerung: pausieren, fortsetzen, beenden	437
13.4.1 Data Race	441
13.5 Interrupt	442
13.6 Warten auf Ereignisse	444
13.7 Reader/Writer-Problem	449
13.7.1 Wenn Threads verhungern	453
13.7.2 Reader/Writer-Varianten	455
13.8 Thread-Sicherheit	456
Teil II: Bausteine komplexer Anwendungen	457
14 Grafische Benutzungsschnittstellen	459
14.1 Ereignisgesteuerte Programmierung	460
14.2 GUI-Programmierung mit Qt	461
14.2.1 Installation und Einsatz	461
14.2.2 Meta-Objektsystem	462
14.2.3 Der Programmablauf	463
14.2.4 Speicher sparen und lokal Daten sichern	464
14.3 Signale, Slots und Widgets	465
14.4 Dialog	473
14.5 Qt oder Boost?	477
14.5.1 Threads	477
14.5.2 Verzeichnisbaum durchwandern	478
15 Internet-Anbindung	481
15.1 Protokolle	482
15.2 Adressen	482
15.3 Socket	486

15.3.1	Bidirektionale Kommunikation.....	489
15.3.2	UDP-Sockets	491
15.3.3	Atomuhr mit UDP abfragen	492
15.4	HTTP	495
15.4.1	Verbindung mit GET	496
15.4.2	Verbindung mit POST	501
15.5	Mini-Webserver	502
16	Datenbankanbindung	511
16.1	C++-Interface.....	512
16.2	Anwendungsbeispiel.....	516
Teil III: Praktische Methoden und Werkzeuge der Softwareentwicklung		523
17	Abläufe automatisieren mit make	525
17.1	Quellen	526
17.2	Wirkungsweise	527
17.3	Variablen und Muster	529
17.4	Universelles Makefile für einfache Projekte	530
18	Unit-Test	533
18.1	Werkzeuge	534
18.2	Test Driven Development	535
18.3	Boost Unit Test Framework.....	536
18.3.1	Beispiel: Testgetriebene Entwicklung einer Operatorfunktion	538
18.3.2	Fixture.....	542
18.3.3	Testprotokoll und Log-Level.....	542
18.3.4	Prüf-Makros	544
18.3.5	Kommandozeilen-Optionen.....	547
19	Werkzeuge zur Verwaltung von Projekten	549
19.1	Dokumentation und Strukturanalyse mit doxygen.....	549
19.1.1	Strukturanalyse.....	553
19.2	Versionskontrolle	554
19.2.1	Subversion	555
19.2.2	Git.....	556
19.3	Projektverwaltung	557

Teil IV: Das C++-Rezeptbuch: Tipps und Lösungen für typische Aufgaben.....559

20	Sichere Programmentwicklung	561
20.1	Regeln zum Design von Methoden	561
20.2	Defensive Programmierung.....	564
20.2.1	double- und float-Werte richtig vergleichen	565
20.2.2	const und constexpr verwenden	565
20.2.3	Anweisungen nach for/if/while einklammern	565
20.2.4	int und unsigned/size_t nicht mischen	566
20.2.5	size_t oder auto statt unsigned int verwenden.....	566
20.2.6	Postfix++ mit Präfix++ implementieren	567
20.2.7	Ein Destruktor darf keine Exception werfen	567
20.2.8	explicit-Typumwandlungsoperator bevorzugen	568
20.2.9	explicit-Konstruktor für eine Typumwandlung bevorzugen	568
20.2.10	Leere Standardkonstruktoren vermeiden	568
20.2.11	Mit override Schreibfehler reduzieren.....	568
20.2.12	Kopieren und Zuweisung verbieten	569
20.2.13	Vererbung verbieten	570
20.2.14	Überschreiben einer virtuellen Methode verhindern	570
20.2.15	The Big Three – oder Big Five?	571
20.2.16	One Definition Rule.....	571
20.2.17	Defensiv Objekte löschen	571
20.2.18	Speicherbeschaffung und -freigabe kapseln.....	572
20.2.19	Programmierrichtlinien einhalten	572
20.3	Exception-sichere Beschaffung von Ressourcen.....	572
20.3.1	Sichere Verwendung von shared_ptr	572
20.3.2	shared_ptr für Arrays korrekt verwenden	573
20.3.3	unique_ptr für Arrays korrekt verwenden	574
20.3.4	Exception-sichere Funktion	574
20.3.5	Exception-sicherer Konstruktor	575
20.3.6	Exception-sichere Zuweisung	578
20.4	Aussagefähige Fehlermeldung ohne neuen String erzeugen.....	579
20.5	Empfehlungen zur Thread-Programmierung.....	581
20.5.1	Warten auf die Freigabe von Ressourcen	581
20.5.2	Deadlock-Vermeidung.....	581
20.5.3	notify_all oder notify_one?.....	582
20.5.4	Performance mit Threads verbessern?.....	583

21	Von der UML nach C++	585
21.1	Vererbung	586
21.2	Interface anbieten und nutzen	586
21.3	Assoziation	588
21.3.1	Aggregation.....	591
21.3.2	Komposition	591
22	Performance, Wert- und Referenzsemantik	593
22.1	Performanceproblem Wertsemantik	595
22.1.1	Auslassen der Kopie	596
22.1.2	Temporäre Objekte bei der Zuweisung	597
22.2	Referenzsemantik für R-Werte	597
22.3	Optimierung durch Referenzsemantik für R-Werte.....	599
22.3.1	Bewegender Konstruktor	602
22.3.2	Bewegender Zuweisungsoperator	603
22.4	Die move()-Funktion	604
22.4.1	Regel zur Template-Auswertung von &&-Parametern	605
22.5	Ein effizienter binärer Plusoperator	606
22.5.1	Kopien temporärer Objekte eliminieren.....	607
22.5.2	Verbesserung durch verzögerte Auswertung	608
23	Effektive Programmierung	611
23.1	Automatische Ermittlung von Abhängigkeiten	612
23.1.1	Getrennte Verzeichnisse: src, obj, bin	613
23.2	Makefile für Verzeichnisbäume	615
23.2.1	Rekursive Make-Aufrufe	616
23.2.2	Ein Makefile für alles	618
23.3	Automatische Erzeugung von Makefiles.....	619
23.3.1	Makefile für rekursive Aufrufe erzeugen	620
23.4	Erzeugen von Bibliotheken	621
23.4.1	Statische Bibliotheksmodule.....	622
23.4.2	Dynamische Bibliotheksmodule.....	623
23.5	GNU Autotools	626
23.6	CMake	629
23.7	Code Bloat bei der Instanziierung von Templates vermeiden.....	629
23.7.1	extern-Template.....	630
23.7.2	Aufspaltung in Schnittstelle und Implementation.....	632

24	Algorithmen für verschiedene Aufgaben	633
24.1	Algorithmen mit Strings	634
24.1.1	String splitten	634
24.1.2	String in Zahl umwandeln	635
24.1.3	Zahl in String umwandeln	639
24.1.4	Strings sprachlich richtig sortieren	639
24.1.5	Umwandlung in Klein- bzw. Großschreibung	641
24.1.6	Strings sprachlich richtig vergleichen	643
24.1.7	Von der Groß-/Kleinschreibung unabhängiger Zeichenvergleich	644
24.1.8	Von der Groß-/Kleinschreibung unabhängige Suche	645
24.2	Textverarbeitung	647
24.2.1	Datei durchsuchen	647
24.2.2	Ersetzungen in einer Datei	648
24.2.3	Code-Formatierer	650
24.2.4	Lines of Code (LOC) ermitteln	651
24.2.5	Zeilen, Wörter und Zeichen einer Datei zählen	653
24.2.6	CSV-Datei lesen	653
24.2.7	Kreuzreferenzliste	655
24.3	Operationen auf Folgen	658
24.3.1	Container anzeigen	658
24.3.2	Folge mit gleichen Werten initialisieren	658
24.3.3	Folge mit Werten eines Generators initialisieren	659
24.3.4	Folge mit fortlaufenden Werten initialisieren	659
24.3.5	Summe und Produkt	660
24.3.6	Mittelwert und Standardabweichung	661
24.3.7	Skalarprodukt	662
24.3.8	Folge der Teilsummen oder -produkte	663
24.3.9	Folge der Differenzen	664
24.3.10	Minimum und Maximum	665
24.3.11	Elemente rotieren	666
24.3.12	Elemente verwürfeln	667
24.3.13	Dubletten entfernen	668
24.3.14	Reihenfolge umdrehen	671
24.3.15	Anzahl der Elemente, die einer Bedingung genügen	671
24.3.16	Gilt X für alle, keins oder wenigstens ein Element einer Folge?	672
24.3.17	Permutationen	674
24.3.18	Lexikografischer Vergleich	676

24.4	Sortieren und Verwandtes	677
24.4.1	Partitionieren	677
24.4.2	Sortieren.....	679
24.4.3	Stabiles Sortieren	680
24.4.4	Partielles Sortieren	681
24.4.5	Das n.-größte oder n.-kleinste Element finden	682
24.4.6	Verschmelzen (merge)	683
24.5	Suchen und Finden	687
24.5.1	Element finden	687
24.5.2	Element einer Menge in der Folge finden	687
24.5.3	Teilfolge finden.....	689
24.5.4	Bestimmte benachbarte Elemente finden	691
24.5.5	Bestimmte aufeinanderfolgende Werte finden	692
24.5.6	Binäre Suche.....	693
24.6	Mengenoperationen auf sortierten Strukturen	695
24.6.1	Teilmengenrelation	696
24.6.2	Vereinigung.....	697
24.6.3	Schnittmenge	697
24.6.4	Differenz	698
24.6.5	Symmetrische Differenz.....	698
24.7	Heap-Algorithmen	699
24.7.1	pop_heap	701
24.7.2	push_heap.....	701
24.7.3	make_heap	702
24.7.4	sort_heap	702
24.7.5	is_heap	703
24.8	Vergleich von Containern auch ungleichen Typs.....	703
24.8.1	Unterschiedliche Elemente finden	703
24.8.2	Prüfung auf gleiche Inhalte	705
24.9	Rechnen mit komplexen Zahlen: Der C++-Standardtyp complex.....	706
24.10	Schnelle zweidimensionale Matrix	708
24.10.1	Optimierung mathematischer Array-Operationen	712
24.11	Singleton.....	717
24.11.1	Implementierung mit einem Zeiger.....	717
24.11.2	Implementierung mit einer Referenz	718
24.11.3	Meyers' Singleton	719
24.12	Vermischtes	721

24.12.1	Erkennung eines Datums.....	721
24.12.2	Erkennung einer IP4-Adresse.....	723
24.12.3	Erzeugen von Zufallszahlen	724
24.12.4	for_each – Auf jedem Element eine Funktion ausführen	728
24.12.5	Verschiedene Möglichkeiten, Container-Bereiche zu kopieren.....	728
24.12.6	Vertauschen von Elementen, Bereichen und Containern	731
24.12.7	Elemente transformieren	731
24.12.8	Ersetzen und Varianten	733
24.12.9	Elemente herausfiltern	734
24.12.10	Grenzwerte von Zahltypen	736
24.12.11	Minimum und Maximum	737
25	Ein- und Ausgabe	739
25.1	Datei- und Verzeichnisoperationen.....	739
25.1.1	Datei oder Verzeichnis löschen.....	740
25.1.2	Datei oder Verzeichnis umbenennen	741
25.1.3	Verzeichnis anlegen.....	742
25.1.4	Verzeichnis anzeigen	743
25.1.5	Verzeichnisbaum anzeigen.....	744
25.2	Tabelle formatiert ausgeben	746
25.3	Formatierte Daten lesen.....	747
25.3.1	Eingabe benutzerdefinierter Typen	747
25.4	Array als Block lesen oder schreiben.....	749
Teil V:	Die C++-Standardbibliothek	751
26	Aufbau und Übersicht	753
26.1	Auslassungen	755
26.2	Beispiele des Buchs und die C++-Standardbibliothek	757
27	Hilfsfunktionen und -klassen	759
27.1	Relationale Operatoren	759
27.2	Unterstützung der Referenzsemantik für R-Werte.....	760
27.2.1	move()	760
27.2.2	forward()	761
27.3	Paare.....	762
27.4	Tupel.....	764
27.5	Funktionsobjekte.....	765

27.5.1	Arithmetische, vergleichende und logische Operationen	765
27.5.2	Funktionsobjekte zum Negieren logischer Prädikate	765
27.5.3	Binden von Argumentwerten	766
27.5.4	Funktionen in Objekte umwandeln	768
27.6	Templates für rationale Zahlen	770
27.7	Hüllklasse für Referenzen	772
27.8	Type Traits	773
27.8.1	Einfache Typ-Abfragen	775
28	Container	777
28.1	Gemeinsame Eigenschaften	779
28.1.1	Initialisierungslisten	781
28.1.2	Konstruktion an Ort und Stelle	781
28.1.3	Reversible Container	782
28.2	Sequenzen	783
28.2.1	vector	784
28.2.2	vector<bool>	785
28.2.3	list	786
28.2.4	deque	788
28.2.5	stack	790
28.2.6	queue	791
28.2.7	priority_queue	792
28.2.8	array	794
28.3	Sortierte assoziative Container	796
28.3.1	map	796
28.3.2	multimap	802
28.3.3	set	802
28.3.4	multiset	805
28.4	Hash-Container	806
28.4.1	unordered_map	808
28.4.2	unordered_multimap	812
28.4.3	unordered_set	812
28.4.4	unordered_multiset	815
28.5	bitset	815
29	Iteratoren	819
29.1	Iterator-Kategorien	820
29.1.1	Anwendung von Traits	822

29.2	distance() und advance()	824
29.3	Reverse-Iteratoren	825
29.4	Insert-Iteratoren.....	826
29.5	Stream-Iteratoren	827
30	Algorithmen	829
30.1	Algorithmen mit Prädikat.....	830
30.1.1	Algorithmen mit binärem Prädikat	830
30.2	Übersicht	831
31	Nationale Besonderheiten	835
31.1	Sprachumgebungen festlegen und ändern	836
31.1.1	Die locale-Funktionen.....	838
31.2	Zeichensätze und -codierung.....	839
31.3	Zeichenklassifizierung und -umwandlung	843
31.4	Kategorien	843
31.4.1	collate.....	843
31.4.2	ctype	844
31.4.3	numeric.....	846
31.4.4	monetary	847
31.4.5	time	850
31.4.6	messages	853
31.5	Konstruktion eigener Facetten	853
32	String	855
33	Speichermanagement	865
33.1	unique_ptr, shared_ptr, weak_ptr.....	865
33.2	new mit Speicherortangabe	869
34	Numerische Arrays (valarray)	871
34.1	Konstruktoren	872
34.2	Elementfunktionen	872
34.3	Binäre Valarray-Operatoren	875
34.4	Mathematische Funktionen.....	877
34.5	slice und slice_array	878
34.6	gslice und gslice_array	881
34.7	mask_array	884
34.8	indirect_array	885

35	Ausgewählte C-Header	887
35.1	<cassert>	888
35.2	<cctype>	888
35.3	<cerrno>	889
35.4	<cmath>.....	889
35.5	<cstdlib>	890
35.6	<cstdlibdef>.....	891
35.7	<cstdio>.....	891
35.8	<cstdliblib>	891
35.9	<cstring>	893
35.10	<ctime>	895
A	Anhang.....	897
A.1	ASCII-Tabelle	897
A.2	C++-Schlüsselwörter	899
A.3	Compilerbefehle	900
A.4	Rangfolge der Operatoren	901
A.5	Lösungen zu den Übungsaufgaben	902
A.6	Installation der Software für Windows	950
A.6.1	Installation des Compilers und der Entwicklungsumgebung.....	950
A.6.2	Integrierte Entwicklungsumgebung einrichten	950
A.6.3	De-Installation.....	952
A.7	Installation der Software für Linux	952
A.7.1	Installation des Compilers	952
A.7.2	Installation von Boost.....	953
A.7.3	Installation und Einrichtung von Code::Blocks.....	954
A.7.4	Beispieldateien entpacken	955
	Glossar.....	957
	Literaturverzeichnis	967
	Register.....	971