

Inhalt

Vorwort	25
Teil I: Einführung in C++	29
1 Es geht los!	31
1.1 Historisches	31
1.2 Objektorientierte Programmierung	32
1.3 Compiler.....	34
1.4 Das erste Programm	35
1.4.1 Namenskonventionen	40
1.5 Integrierte Entwicklungsumgebung	41
1.6 Einfache Datentypen und Operatoren	43
1.6.1 Ausdruck	43
1.6.2 Ganze Zahlen	43
1.6.3 Reelle Zahlen	49
1.6.4 Konstante	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	71
1.8.5	Wiederholungen	73
1.8.6	Kontrolle mit break und continue	80
1.9	Benutzerdefinierte und zusammengesetzte Datentypen	83
1.9.1	Aufzählungstypen	83
1.9.2	Strukturen	85
1.9.3	Der C++-Standardtyp vector	87
1.9.4	Zeichenketten: Der C++-Standardtyp string	91
1.9.5	Container und Schleifen	94
1.9.6	Typermittlung mit auto	95
1.9.7	Unions und Bitfelder	96
1.10	Einfache Ein- und Ausgabe	98
1.10.1	Standardein- und -ausgabe	98
1.10.2	Ein- und Ausgabe mit Dateien	102
2	Programmstrukturierung	107
2.1	Funktionen	108
2.1.1	Aufbau und Prototypen	108
2.1.2	Gültigkeitsbereiche und Sichtbarkeit in Funktionen	110
2.1.3	Lokale static-Variable: Funktion mit Gedächtnis	111
2.2	Schnittstellen zum Datentransfer	112
2.2.1	Übergabe per Wert	113
2.2.2	Übergabe per Referenz	117
2.2.3	Gefahren bei der Rückgabe von Referenzen	118
2.2.4	Vorgegebene Parameterwerte und unterschiedliche Parameterzahl	119
2.2.5	Überladen von Funktionen	120
2.2.6	Funktion main()	121
2.2.7	Beispiel Taschenrechnersimulation	122
2.2.8	Spezifikation von Funktionen	127
2.2.9	Alternative Funktions-Syntax	127
2.3	Modulare Programmgestaltung	127
2.3.1	Steuerung der Übersetzung nur mit #include	128
2.3.2	Einbinden vorübersetzter Programmteile	128

2.3.3	Dateiübergreifende Gültigkeit und Sichtbarkeit.....	130
2.3.4	Übersetzungseinheit, Deklaration, Definition	132
2.3.5	Präprozessordirektiven und Makros	134
2.4	Funktions-Templates	140
2.4.1	Spezialisierung von Templates	143
2.4.2	Einbinden von Templates	144
2.5	inline-Funktionen.....	146
2.6	constexpr-Funktionen.....	146
2.7	Namensräume	149
2.8	C++-Header	150
2.8.1	Einbinden von C-Funktionen	152
3	Objektorientierung 1	153
3.1	Abstrakte Datentypen	154
3.2	Klassen und Objekte	155
3.2.1	const-Objekte und Methoden.....	159
3.2.2	inline-Elementfunktionen.....	159
3.3	Initialisierung und Konstruktoren	161
3.3.1	Standardkonstruktor.....	161
3.3.2	Direkte Initialisierung der Attribute	162
3.3.3	Allgemeine Konstruktoren	162
3.3.4	Kopierkonstruktor	165
3.3.5	Typumwandlungskonstruktor	168
3.3.6	Konstruktor und mehr vorgeben oder verbieten.....	170
3.3.7	Einheitliche Initialisierung und Sequenzkonstruktor.....	170
3.3.8	Delegierender Konstruktor	172
3.3.9	constexpr-Konstruktor und -Methoden	173
3.4	Beispiel: Rationale Zahlen	176
3.4.1	Aufgabenstellung	176
3.4.2	Entwurf.....	178
3.4.3	Implementation.....	181
3.5	Destruktoren	186
3.6	Wie kommt man zu Klassen und Objekten? Ein Beispiel.....	188
3.7	Gegenseitige Abhängigkeit von Klassen	193
4	Intermezzo: Zeiger	195
4.1	Zeiger und Adressen	196
4.2	C-Arrays.....	199

4.2.1	C-Array und sizeof.....	201
4.2.2	Initialisierung von C-Arrays.....	202
4.2.3	Zeigerarithmetik.....	202
4.2.4	Indexoperator bei C-Arrays.....	203
4.2.5	C-Array mit begin() und end() durchlaufen	203
4.3	C-Zeichenketten.....	204
4.4	Dynamische Datenobjekte	211
4.4.1	Freigeben dynamischer Objekte.....	213
4.5	Zeiger und Funktionen	216
4.5.1	Parameterübergabe mit Zeigern.....	216
4.5.2	Parameter des main-Programms	218
4.5.3	Gefahren bei der Rückgabe von Zeigern.....	219
4.6	this-Zeiger	220
4.7	Mehrdimensionale C-Arrays.....	220
4.7.1	Statische mehrdimensionale C-Arrays	220
4.7.2	Array als Funktionsparameter.....	221
4.7.3	Dynamisch erzeugte mehrdimensionale Arrays.....	225
4.7.4	Klasse für dynamisches zweidimensionales Array	227
4.8	Binäre Ein-/Ausgabe	233
4.9	Zeiger auf Funktionen.....	236
4.10	Standard-Typumwandlungen für Zeiger.....	240
4.11	Zeiger auf Elementfunktionen und -daten	241
4.11.1	Zeiger auf Elementfunktionen.....	241
4.11.2	Zeiger auf Elementdaten	242
4.12	Komplexe Deklarationen lesen	242
4.12.1	Lesbarkeit mit typedef und using verbessern	243
5	Objektorientierung 2	245
5.1	Eine String-Klasse	245
5.1.1	friend-Funktionen	251
5.2	Klassenspezifische Daten und Funktionen	252
5.2.1	Klassenspezifische Konstante.....	256
5.3	Klassen-Templates	258
5.3.1	Ein Stack-Template	258
5.3.2	Stack mit statisch festgelegter Größe.....	260
5.4	Template-Metaprogrammierung	262
5.5	Variadic Templates: Templates mit variabler Parameterzahl	265

5.5.1	Klassen-Templates mit variabler Stelligkeit	268
5.6	Typbestimmung mit decltype	269
5.6.1	decltype	272
6	Vererbung	273
6.1	Vererbung und Initialisierung	279
6.2	Zugriffsschutz	280
6.3	Typbeziehung zwischen Ober- und Unterklasse	282
6.4	Code-Wiederverwendung	283
6.4.1	Konstruktor erben	284
6.5	Überschreiben von Funktionen in abgeleiteten Klassen	286
6.5.1	Virtuelle Funktionen	287
6.5.2	Abstrakte Klassen	292
6.5.3	Virtueller Destruktor	297
6.5.4	Private virtuelle Funktionen	300
6.6	Probleme der Modellierung mit Vererbung	302
6.7	Mehrfachvererbung	305
6.7.1	Namenskonflikte	307
6.7.2	Virtuelle Basisklassen	309
6.8	Standard-Typumwandlungsoperatoren	312
6.9	Typinformationen zur Laufzeit	315
6.10	Using-Deklaration für protected-Funktionen	316
6.11	Private- und Protected-Vererbung	317
7	Fehlerbehandlung	321
7.1	Ausnahmebehandlung	323
7.1.1	Exception-Spezifikation in Deklarationen	326
7.1.2	Exception-Hierarchie in C++	327
7.1.3	Besondere Fehlerbehandlungsfunktionen	328
7.1.4	Erkennen logischer Fehler	330
7.1.5	Arithmetische Fehler / Division durch 0	332
7.2	Speicherbeschaffung mit new	333
7.3	Exception-Sicherheit	335
8	Überladen von Operatoren	337
8.1	Rationale Zahlen – noch einmal	339
8.1.1	Arithmetische Operatoren	339
8.1.2	Ausgabeoperator <<	342

8.2	Eine Klasse für Vektoren	343
8.2.1	Index-Operator []	346
8.2.2	Zuweisungsoperator =	349
8.2.3	Mathematische Vektoren	351
8.2.4	Multiplikationsoperator	353
8.3	Inkrement-Operator ++	354
8.4	Typumwandlungsoperator	358
8.5	Smart Pointer: Operatoren -> und *	359
8.5.1	Smart Pointer und die C++-Standardbibliothek	364
8.6	Objekt als Funktion	364
8.7	new und delete überladen	366
8.7.1	Speichermanagement mit malloc und free	370
8.7.2	Unterscheidung zwischen Heap- und Stack-Objekten	371
8.7.3	Fehlende delete-Anweisung entdecken	373
8.7.4	Eigene Speicherverwaltung für einen bestimmten Typ	374
8.7.5	Empfehlungen im Umgang mit new und delete	378
8.8	Operatoren für Literale	379
8.8.1	Stringlitterale	379
8.8.2	Benutzerdefinierte Literale	380
8.9	Mehrdimensionale Matrizen	383
8.9.1	Zweidimensionale Matrix als Vektor von Vektoren	384
8.9.2	Dreidimensionale Matrix	386
8.10	Zuweisung bei Vererbung	389
9	Dateien und Ströme	399
9.1	Ausgabe	401
9.1.1	Formatierung der Ausgabe	401
9.2	Eingabe	404
9.3	Manipulatoren	407
9.3.1	Eigene Manipulatoren	412
9.4	Fehlerbehandlung	414
9.5	Typumwandlung von Dateioobjekten nach bool	415
9.6	Arbeit mit Dateien	416
9.6.1	Positionierung in Dateien	417
9.6.2	Lesen und Schreiben in derselben Datei	418
9.7	Umleitung auf Strings	419
9.8	Tabelle formatiert ausgeben	421

9.9	Formatierte Daten lesen	422
9.9.1	Eingabe benutzerdefinierter Typen	422
9.10	Blockweise lesen oder schreiben	423
9.11	Ergänzungen.....	426
10	Einführung in die Standard Template Library (STL)	427
10.1	Container, Iteratoren, Algorithmen	428
10.2	Iteratoren im Detail	433
10.3	Beispiel verkettete Liste	434
Teil II:	Fortgeschrittene Themen	439
11	Lambda-Funktionen.....	441
11.1	Eigenschaften.....	442
11.1.1	Äquivalenz zum Funktionszeiger	443
11.1.2	Lambda-Funktion und Klasse	444
11.2	Generische Lambda-Funktionen	444
11.3	Parametererfassung mit []	446
12	Performance, Wert- und Referenzsemantik	447
12.1	Performanceproblem Wertsemantik	449
12.1.1	Auslassen der Kopie	449
12.1.2	Temporäre Objekte bei der Zuweisung	450
12.2	Referenzsemantik für R-Werte	451
12.3	Optimierung durch Referenzsemantik für R-Werte.....	453
12.3.1	Bewegender Konstruktor	456
12.3.2	Bewegender Zuweisungsoperator	456
12.4	Die move()-Funktion	457
12.4.1	Regel zur Template-Auswertung von &&t-Parametern	459
12.5	Ein effizienter binärer Plusoperator	460
12.5.1	Return Value Optimization (RVO).....	460
12.5.2	Kopien temporärer Objekte eliminieren	461
12.5.3	Verbesserung durch verzögerte Auswertung	462
13	Reguläre Ausdrücke	465
13.1	Elemente regulärer Ausdrücke	466
13.1.1	Greedy oder lazy?	468
13.2	Interaktive Auswertung	469

13.3 Auszug des regex-APIs	472
13.4 Anwendungen	474
14 Threads	475
14.1 Zeit und Dauer	476
14.2 Threads	477
14.3 Die Klasse thread	480
14.3.1 Thread-Group	483
14.4 Synchronisation	485
14.5 Thread-Steuerung: pausieren, fortsetzen, beenden	488
14.5.1 Data Race	492
14.6 Interrupt	493
14.7 Warten auf Ereignisse	495
14.8 Reader/Writer-Problem	500
14.8.1 Wenn Threads verhungern	505
14.8.2 Reader/Writer-Varianten	507
14.9 Thread-Sicherheit	507
15 Grafische Benutzungsschnittstellen	509
15.1 Ereignisgesteuerte Programmierung	510
15.2 GUI-Programmierung mit Qt	511
15.2.1 Installation und Einsatz	511
15.2.2 Meta-Objektsystem	512
15.2.3 Der Programmablauf	513
15.2.4 Speicher sparen und lokal Daten sichern	514
15.3 Signale, Slots und Widgets	516
15.4 Dialog	524
15.5 Qt oder Boost?	527
15.5.1 Threads	527
15.5.2 Verzeichnisbaum durchwandern	529
16 Internet-Anbindung	531
16.1 Protokolle	532
16.2 Adressen	532
16.3 Socket	536
16.3.1 Bidirektionale Kommunikation	539
16.3.2 UDP-Sockets	541
16.3.3 Atomuhr mit UDP abfragen	542

16.4	HTTP	545
16.4.1	Verbindung mit GET	546
16.4.2	Verbindung mit POST	551
16.5	Mini-Webserver	552
17	Datenbankanbindung	561
17.1	C++-Interface	562
17.2	Anwendungsbeispiel	566
 Teil III: Praktische Methoden und Werkzeuge der Softwareentwicklung		573
18	Effiziente Programmerzeugung mit make	575
18.1	Quellen	576
18.2	Wirkungsweise	577
18.3	Variablen und Muster	579
18.4	Universelles Makefile für einfache Projekte	580
18.5	Automatische Ermittlung von Abhängigkeiten	582
18.5.1	Getrennte Verzeichnisse: src, obj, bin	583
18.6	Makefile für Verzeichnisbäume	585
18.6.1	Rekursive Make-Aufrufe	586
18.6.2	Ein Makefile für alles	588
18.7	Automatische Erzeugung von Makefiles	589
18.7.1	Makefile für rekursive Aufrufe erzeugen	590
18.8	Erzeugen von Bibliotheken	591
18.8.1	Statische Bibliotheksmodule	592
18.8.2	Dynamische Bibliotheksmodule	593
18.9	GNU Autotools	596
18.10	CMake	599
18.11	Code Bloat bei der Instanziierung von Templates vermeiden	599
18.11.1	extern-Template	600
18.11.2	Aufspaltung in Schnittstelle und Implementation	602
19	Unit-Test	603
19.1	Werkzeuge	604
19.2	Test Driven Development	605
19.3	Boost Unit Test Framework	606
19.3.1	Beispiel: Testgetriebene Entwicklung einer Operatorfunktion	608

19.3.2	Fixture	612
19.3.3	Testprotokoll und Log-Level	613
19.3.4	Prüf-Makros	614
19.3.5	Kommandozeilen-Optionen	617
20	Werkzeuge zur Verwaltung von Projekten	619
20.1	Dokumentation und Strukturanalyse mit doxygen	619
20.1.1	Strukturanalyse	623
20.2	Versionskontrolle	624
20.2.1	Subversion	625
20.2.2	Git	626
20.3	Projektverwaltung	627
Teil IV: Das C++-Rezeptbuch: Tipps und Lösungen für typische Aufgaben		629
21	Sichere Programmentwicklung	631
21.1	Regeln zum Design von Methoden	631
21.2	Defensive Programmierung	634
21.2.1	double- und float-Werte richtig vergleichen	635
21.2.2	const und constexpr verwenden	635
21.2.3	Anweisungen nach for/if/while einklammern	635
21.2.4	int und unsigned/size_t nicht mischen	636
21.2.5	size_t oder auto statt unsigned int verwenden	636
21.2.6	Postfix++ mit Präfix++ implementieren	637
21.2.7	Ein Destruktor darf keine Exception werfen	637
21.2.8	explicit-Typumwandlungsoperator bevorzugen	638
21.2.9	explicit-Konstruktor für eine Typumwandlung bevorzugen	638
21.2.10	Leere Standardkonstruktoren vermeiden	638
21.2.11	Mit override Schreibfehler reduzieren	638
21.2.12	Kopieren und Zuweisung verbieten	639
21.2.13	Vererbung verbieten	640
21.2.14	Überschreiben einer virtuellen Methode verhindern	640
21.2.15	The Big Three – oder Big Five?	641
21.2.16	One Definition Rule	642
21.2.17	Defensiv Objekte löschen	642
21.2.18	Speicherbeschaffung und -freigabe kapseln	642

21.2.19	Programmierrichtlinien einhalten	642
21.3	Exception-sichere Beschaffung von Ressourcen.....	642
21.3.1	Sichere Verwendung von <code>unique_ptr</code> und <code>shared_ptr</code>	643
21.3.2	So vermeiden Sie <code>new</code> und <code>delete</code> !.....	643
21.3.3	So vermeiden Sie <code>new[]</code> und <code>delete[]</code> !	644
21.3.4	<code>shared_ptr</code> für Arrays korrekt verwenden	644
21.3.5	<code>unique_ptr</code> für Arrays korrekt verwenden	645
21.3.6	Exception-sichere Funktion	645
21.3.7	Exception-sicherer Konstruktor	646
21.3.8	Exception-sichere Zuweisung	649
21.4	Empfehlungen zur Thread-Programmierung	650
21.4.1	Warten auf die Freigabe von Ressourcen	650
21.4.2	Deadlock-Vermeidung	651
21.4.3	<code>notify_all</code> oder <code>notify_one</code> ?.....	651
21.4.4	Performance mit Threads verbessern?.....	652
22	Von der UML nach C++	653
22.1	Vererbung	654
22.2	Interface anbieten und nutzen	654
22.3	Assoziation	656
22.3.1	Aggregation.....	659
22.3.2	Komposition	659
23	Algorithmen für verschiedene Aufgaben	661
23.1	Algorithmen mit Strings	662
23.1.1	String splitten	662
23.1.2	String in Zahl umwandeln.....	663
23.1.3	Zahl in String umwandeln	667
23.1.4	Strings sprachlich richtig sortieren	667
23.1.5	Umwandlung in Klein- bzw. Großschreibung	669
23.1.6	Strings sprachlich richtig vergleichen.....	671
23.1.7	Von der Groß-/Kleinschreibung unabhängiger Zeichenvergleich	672
23.1.8	Von der Groß-/Kleinschreibung unabhängige Suche	673
23.2	Textverarbeitung	675
23.2.1	Datei durchsuchen	675
23.2.2	Ersetzungen in einer Datei	676
23.2.3	Code-Formatierer	678
23.2.4	Lines of Code (LOC) ermitteln	679

23.2.5	Zeilen, Wörter und Zeichen einer Datei zählen	681
23.2.6	CSV-Datei lesen	681
23.2.7	Kreuzreferenzliste	683
23.3	Operationen auf Folgen	686
23.3.1	Container anzeigen	686
23.3.2	Folge mit gleichen Werten initialisieren	686
23.3.3	Folge mit Werten eines Generators initialisieren	687
23.3.4	Folge mit fortlaufenden Werten initialisieren	687
23.3.5	Summe und Produkt	688
23.3.6	Mittelwert und Standardabweichung	689
23.3.7	Skalarprodukt	689
23.3.8	Folge der Teilsummen oder -produkte	690
23.3.9	Folge der Differenzen	691
23.3.10	Kleinstes und größtes Element finden	692
23.3.11	Elemente rotieren	694
23.3.12	Elemente verwürfeln	695
23.3.13	Dubletten entfernen	696
23.3.14	Reihenfolge umdrehen	698
23.3.15	Anzahl der Elemente, die einer Bedingung genügen	699
23.3.16	Gilt X für alle, keins oder wenigstens ein Element einer Folge?	700
23.3.17	Permutationen	701
23.3.18	Lexikografischer Vergleich	703
23.4	Sortieren und Verwandtes	705
23.4.1	Partitionieren	705
23.4.2	Sortieren	707
23.4.3	Stabiles Sortieren	708
23.4.4	Partielles Sortieren	709
23.4.5	Das n.-größte oder n.-kleinste Element finden	710
23.4.6	Verschmelzen (merge)	711
23.5	Suchen und Finden	714
23.5.1	Element finden	714
23.5.2	Element einer Menge in der Folge finden	715
23.5.3	Teilfolge finden	716
23.5.4	Bestimmte benachbarte Elemente finden	718
23.5.5	Bestimmte aufeinanderfolgende Werte finden	719
23.5.6	Binäre Suche	720
23.6	Mengenoperationen auf sortierten Strukturen	723

23.6.1	Teilmengenrelation	723
23.6.2	Vereinigung	724
23.6.3	Schnittmenge	725
23.6.4	Differenz	725
23.6.5	Symmetrische Differenz	726
23.7	Heap-Algorithmen	727
23.7.1	pop_heap	728
23.7.2	push_heap	729
23.7.3	make_heap	729
23.7.4	sort_heap	730
23.7.5	is_heap	730
23.8	Vergleich von Containern auch ungleichen Typs	731
23.8.1	Unterschiedliche Elemente finden	731
23.8.2	Prüfung auf gleiche Inhalte	733
23.9	Rechnen mit komplexen Zahlen: Der C++-Standardtyp complex	734
23.10	Schnelle zweidimensionale Matrix	736
23.10.1	Optimierung mathematischer Array-Operationen	739
23.11	Singleton	742
23.11.1	Implementierung mit einem Zeiger	742
23.11.2	Implementierung mit einer Referenz	743
23.11.3	Meyers' Singleton	744
23.12	Vermischtes	746
23.12.1	Erkennung eines Datums	746
23.12.2	Erkennung einer IP4-Adresse	748
23.12.3	Erzeugen von Zufallszahlen	749
23.12.4	for_each – Auf jedem Element eine Funktion ausführen	754
23.12.5	Verschiedene Möglichkeiten, Container-Bereiche zu kopieren	754
23.12.6	Vertauschen von Elementen, Bereichen und Containern	757
23.12.7	Elemente transformieren	757
23.12.8	Ersetzen und Varianten	759
23.12.9	Elemente herausfiltern	760
23.12.10	Grenzwerte von Zahltypen	762
23.12.11	Minimum und Maximum	763
24	Datei- und Verzeichnisoperationen	765
24.1	Datei oder Verzeichnis löschen	766
24.2	Datei oder Verzeichnis umbenennen	767

24.3	Verzeichnis anlegen	768
24.4	Verzeichnis anzeigen	769
24.5	Verzeichnisbaum anzeigen	770
Teil V: Die C++-Standardbibliothek		773
25	Aufbau und Übersicht	775
25.1	Auslassungen	777
25.2	Beispiele des Buchs und die C++-Standardbibliothek	779
26	Hilfsfunktionen und -klassen	781
26.1	Relationale Operatoren	781
26.2	Unterstützung der Referenzsemantik für R-Werte	782
26.2.1	move()	782
26.2.2	forward()	783
26.3	Paare	784
26.4	Tupel	786
26.5	Indexfolgen	788
26.6	Funktionsobjekte	789
26.6.1	Arithmetische, vergleichende und logische Operationen	789
26.6.2	Funktionsobjekte zum Negieren logischer Prädikate	790
26.6.3	Binden von Argumentwerten	791
26.6.4	Funktionen in Objekte umwandeln	792
26.7	Templates für rationale Zahlen	794
26.8	Hüllklasse für Referenzen	796
26.9	Type Traits	797
26.9.1	Wie funktionieren Type Traits? – ein Beispiel	798
26.9.2	Abfrage von Eigenschaften	800
26.9.3	Abfrage numerischer Eigenschaften	802
26.9.4	Typbeziehungen	802
26.9.5	Typumwandlungen	802
26.9.6	Auswahl weiterer Transformationen	803
27	Container	805
27.1	Gemeinsame Eigenschaften	807
27.1.1	Initialisierungslisten	809
27.1.2	Konstruktion an Ort und Stelle	810
27.1.3	Reversible Container	810

27.2	Sequenzen	811
27.2.1	vector	812
27.2.2	vector<bool>	813
27.2.3	list	814
27.2.4	deque	817
27.2.5	stack	818
27.2.6	queue	820
27.2.7	priority_queue	821
27.2.8	array	823
27.3	Sortierte assoziative Container	826
27.3.1	map	826
27.3.2	multimap	831
27.3.3	set	831
27.3.4	multiset	835
27.4	Hash-Container	835
27.4.1	unordered_map	837
27.4.2	unordered_multimap	842
27.4.3	unordered_set	843
27.4.4	unordered_multiset	845
27.5	bitset	846
28	Iteratoren	849
28.1	Iterator-Kategorien	850
28.1.1	Anwendung von Traits	852
28.2	distance() und advance()	854
28.3	Bereichszugriff	855
28.3.1	Reverse-Iteratoren	856
28.4	Insert-Iteratoren	857
28.5	Stream-Iteratoren	858
29	Algorithmen	861
29.1	Algorithmen mit Prädikat	862
29.1.1	Algorithmen mit binärem Prädikat	862
29.2	Übersicht	863
30	Nationale Besonderheiten	867
30.1	Sprachumgebungen festlegen und ändern	868
30.1.1	Die locale-Funktionen	870

30.2	Zeichensätze und -codierung.....	871
30.3	Zeichenklassifizierung und -umwandlung	875
30.4	Kategorien	876
30.4.1	collate.....	876
30.4.2	ctype	877
30.4.3	numeric.....	878
30.4.4	monetary	880
30.4.5	time	883
30.4.6	messages	885
30.5	Konstruktion eigener Facetten	886
31	String	889
32	Speichermanagement	899
32.1	unique_ptr	899
32.1.1	make_unique	901
32.2	shared_ptr.....	902
32.2.1	make_shared	903
32.3	weak_ptr	904
32.4	new mit Speicherortangabe	905
33	Numerische Arrays (valarray)	907
33.1	Konstruktoren	908
33.2	Elementfunktionen	908
33.3	Binäre Valarray-Operatoren	911
33.4	Mathematische Funktionen.....	913
33.5	slice und slice_array	914
33.6	gslice und gslice_array.....	917
33.7	mask_array	920
33.8	indirect_array	921
34	Ausgewählte C-Header	923
34.1	<cassert>	924
34.2	<cctype>	924
34.3	<errno>	925
34.4	<cmath>.....	925
34.5	<cstdint>	926
34.6	<cstdint>.....	927

34.7 <cstdio>	927
34.8 <cstdlib>	927
34.9 <cstring>	929
34.10 <ctime>	931
A Anhang	933
A.1 ASCII-Tabelle	933
A.2 C++-Schlüsselwörter	935
A.3 Compilerbefehle	936
A.4 Rangfolge der Operatoren	937
A.5 Lösungen zu den Übungsaufgaben	938
A.6 Installation der Software für Windows	948
A.6.1 Installation des Compilers und der Entwicklungsumgebung	948
A.6.2 Integrierte Entwicklungsumgebung einrichten	949
A.6.3 De-Installation	950
A.7 Installation der Software für Linux	951
A.7.1 Installation des Compilers	951
A.7.2 Installation von Boost	952
A.7.3 Installation und Einrichtung von Code::Blocks	952
A.7.4 Beispieldateien entpacken	953
Glossar	955
Literaturverzeichnis	965
Register	969