

# Inhalt

<b>Vorwort .....</b>	<b>21</b>
<b>Teil I: Einführung in C++ .....</b>	<b>25</b>
<b>1 Es geht los! .....</b>	<b>27</b>
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
1.5.3 KDevelop .....	41
1.6 Einfache Datentypen und Operatoren .....	43
1.6.1 Ausdruck .....	44
1.6.2 Ganze Zahlen .....	44
1.6.3 Reelle Zahlen .....	48
1.6.4 Konstante .....	52
1.6.5 Zeichen .....	53
1.6.6 Logischer Datentyp bool .....	56
1.6.7 Referenzen .....	57
1.6.8 Regeln zum Bilden von Ausdrücken .....	58

1.6.9	Standard-Typumwandlungen .....	59
1.7	Gültigkeitsbereich und Sichtbarkeit .....	60
1.7.1	Namespace std.....	61
1.8	Kontrollstrukturen .....	62
1.8.1	Anweisungen .....	62
1.8.2	Sequenz (Reihung) .....	64
1.8.3	Auswahl (Selektion, Verzweigung) .....	64
1.8.4	Fallunterscheidungen mit switch .....	69
1.8.5	Wiederholungen.....	71
1.8.6	Kontrolle mit break und continue .....	78
1.9	Benutzerdefinierte und zusammengesetzte Datentypen .....	80
1.9.1	Aufzählungstypen .....	80
1.9.2	Arrays: Der C++-Standardtyp vector.....	82
1.9.3	Zeichenketten: Der C++-Standardtyp string .....	87
1.9.4	Strukturierte Datentypen.....	89
<b>2</b>	<b>Einfache Ein- und Ausgabe .....</b>	<b>93</b>
2.1	Standardein- und -ausgabe.....	93
2.2	Ein- und Ausgabe mit Dateien .....	96
<b>3</b>	<b>Programmstrukturierung .....</b>	<b>101</b>
3.1	Funktionen.....	102
3.1.1	Aufbau und Prototypen .....	102
3.1.2	Gültigkeitsbereiche und Sichtbarkeit in Funktionen .....	104
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.....	123
3.3.2	Einbinden vorübersetzter Programmteile .....	123
3.3.3	Dateiübergreifende Gültigkeit und Sichtbarkeit.....	125
3.3.4	Übersetzungseinheit, Deklaration, Definition .....	127

3.3.5	Compilerdirektiven und Makros .....	129
3.4	Funktions-Templates .....	135
3.4.1	Spezialisierung von Templates .....	138
3.4.2	Einbinden von Templates .....	139
3.5	inline-Funktionen.....	140
3.6	Namensräume .....	142
3.7	C++-Header.....	143
3.7.1	Einbinden von C-Funktionen .....	145
<b>4</b>	<b>Objektorientierung 1 .....</b>	<b>147</b>
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 .....	179
4.9	Delegierender Konstruktor .....	181
<b>5</b>	<b>Intermezzo: Zeiger .....</b>	<b>183</b>
5.1	Zeiger und Adressen.....	184
5.2	C-Arrays.....	187
5.2.1	C-Arrays und sizeof .....	189
5.2.2	Indexoperator bei C-Arrays.....	189
5.2.3	Initialisierung von C-Arrays.....	190
5.2.4	Zeigerarithmetik.....	190
5.3	C-Zeichenketten.....	191

5.4	Dynamische Datenobjekte .....	198
5.4.1	Freigeben dynamischer Objekte .....	201
5.5	Zeiger und Funktionen .....	203
5.5.1	Parameterübergabe mit Zeigern .....	203
5.5.2	Parameter des main-Programms .....	205
5.5.3	Gefahren bei der Rückgabe von Zeigern .....	206
5.6	Mehrdimensionale C-Arrays .....	207
5.6.1	Statische mehrdimensionale C-Arrays .....	207
5.6.2	Dynamisch erzeugte mehrdimensionale Arrays .....	211
5.6.3	Klasse für dynamisches zweidimensionales Array .....	213
5.7	Binäre Ein-/Ausgabe .....	217
5.8	Zeiger auf Funktionen .....	220
5.9	this-Zeiger .....	223
5.10	Komplexe Deklarationen lesen .....	224
5.11	Standard-Typumwandlungen für Zeiger .....	226
5.12	Zeiger auf Elementfunktionen und -daten .....	227
5.12.1	Zeiger auf Elementfunktionen .....	227
5.12.2	Zeiger auf Elementdaten .....	228
<b>6</b>	<b>Objektorientierung 2 .....</b>	<b>229</b>
6.1	Eine String-Klasse .....	229
6.1.1	Optimierung der Klasse MeinString .....	234
6.1.2	friend-Funktionen .....	236
6.2	Klassenspezifische Daten und Funktionen .....	238
6.2.1	Klassenspezifische Konstante .....	241
6.3	Klassen-Templates .....	242
6.3.1	Ein Stack-Template .....	242
6.3.2	Stack mit statisch festgelegter Größe .....	245
6.4	Template-Metaprogrammierung .....	247
6.5	Variadic Templates: Templates mit variabler Parameterzahl .....	249
<b>7</b>	<b>Vererbung .....</b>	<b>253</b>
7.1	Vererbung und Initialisierung .....	259
7.2	Zugriffsschutz .....	260
7.3	Typbeziehung zwischen Ober- und Unterklasse .....	262
7.4	Code-Wiederverwendung .....	263
7.5	Überschreiben von Funktionen in abgeleiteten Klassen .....	264
7.6	Polymorphismus .....	266

7.6.1	Virtuelle Funktionen.....	266
7.6.2	Abstrakte Klassen .....	271
7.6.3	Virtueller Destruktor.....	276
7.7	Probleme der Modellierung mit Vererbung.....	278
7.8	Mehrfachvererbung.....	281
7.8.1	Namenskonflikte .....	284
7.8.2	Virtuelle Basisklassen .....	285
7.9	Standard-Typumwandlungsoperatoren .....	288
7.10	Typinformationen zur Laufzeit.....	291
7.11	Using-Deklaration für Klassen .....	292
7.12	Private- und Protected-Vererbung.....	293
<b>8</b>	<b>Fehlerbehandlung.....</b>	<b>297</b>
8.1	Ausnahmebehandlung .....	299
8.1.1	Exception-Spezifikation in Deklarationen.....	302
8.1.2	Exception-Hierarchie in C++ .....	303
8.1.3	Besondere Fehlerbehandlungsfunktionen.....	305
8.1.4	Erkennen logischer Fehler .....	306
8.2	Speicherbeschaffung mit new .....	308
8.3	Exception-Sicherheit .....	311
<b>9</b>	<b>Überladen von Operatoren .....</b>	<b>313</b>
9.1	Rationale Zahlen – noch einmal .....	315
9.1.1	Arithmetische Operatoren.....	315
9.1.2	Ausgabeoperator << .....	318
9.2	Eine Klasse für Vektoren .....	319
9.2.1	Index-Operator [ ].....	322
9.2.2	Zuweisungsoperator = .....	324
9.2.3	Zuweisungsoperator und Vererbung.....	326
9.2.4	Mathematische Vektoren .....	330
9.2.5	Multiplikationsoperator .....	332
9.3	Inkrement-Operator ++ .....	333
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
9.7	new und delete überladen .....	346
9.7.1	Speichermanagement mit malloc und free .....	349

9.7.2	Unterscheidung zwischen Heap- und Stack-Objekten .....	351
9.7.3	Fehlende delete-Anweisung entdecken .....	352
9.7.4	Eigene Speicherverwaltung .....	353
9.7.5	Empfehlungen im Umgang mit new und delete .....	357
9.8	Mehrdimensionale Matrizen .....	358
9.8.1	Zweidimensionale Matrix als Vektor von Vektoren .....	359
9.8.2	Dreidimensionale Matrix .....	362
<b>10</b>	<b>Dateien und Ströme .....</b>	<b>365</b>
10.1	Ausgabe .....	367
10.1.1	Formatierung der Ausgabe .....	367
10.2	Eingabe .....	370
10.3	Manipulatoren .....	373
10.3.1	Eigene Manipulatoren .....	376
10.4	Fehlerbehandlung .....	377
10.4.1	Exception ios::failure .....	379
10.5	Typumwandlung von Dateiobjekten nach bool .....	379
10.6	Arbeit mit Dateien .....	380
10.6.1	Positionierung in Dateien .....	381
10.6.2	Lesen und Schreiben in derselben Datei .....	382
10.7	Umleitung auf Strings .....	383
10.8	Ergänzungen .....	385
<b>11</b>	<b>Einführung in die Standard Template Library (STL) .....</b>	<b>387</b>
11.1	Das Konzept: Container, Iteratoren, Algorithmen .....	389
11.2	Iteratoren im Detail .....	393
11.3	Beispiel verkettete Liste .....	394
<b>12</b>	<b>Reguläre Ausdrücke .....</b>	<b>399</b>
12.1	Elemente regulärer Ausdrücke .....	400
12.1.1	Greedy oder lazy? .....	402
12.2	Interaktive Auswertung .....	403
12.3	Auszug des regex-APIs .....	406
12.4	Anwendungen .....	408
<b>13</b>	<b>Threads .....</b>	<b>409</b>
13.1	Die Klasse thread .....	413
13.2	Synchronisation .....	416

13.2.1	Thread-Group.....	418
13.3	Thread-Steuerung: pausieren, fortsetzen, beenden.....	419
13.4	Interrupt.....	424
13.5	Warten auf Ereignisse .....	426
13.6	Reader/Writer-Problem.....	431
13.6.1	Wenn Threads verhungern.....	435
13.6.2	Reader/Writer-Varianten .....	437
13.7	Thread-Sicherheit .....	438

## **Teil II: Bausteine komplexer Anwendungen .....439**

### **14 Grafische Benutzungsschnittstellen ..... 441**

14.1	Ereignisgesteuerte Programmierung.....	442
14.2	GUI-Programmierung mit Qt.....	443
14.2.1	Meta-Objektsystem .....	443
14.2.2	Der Programmablauf .....	444
14.2.3	Speicher sparen und lokal Daten sichern .....	445
14.3	Signale, Slots und Widgets .....	447
14.4	Dialog .....	455
14.5	Qt oder Boost?.....	458
14.5.1	Threads .....	459
14.5.2	Verzeichnisbaum durchwandern .....	460

### **15 Internet-Anbindung .....463**

15.1	Protokolle .....	464
15.2	Adressen.....	464
15.3	Socket .....	468
15.3.1	Bidirektionale Kommunikation.....	471
15.3.2	UDP-Sockets .....	473
15.3.3	Atomuhr mit UDP abfragen .....	474
15.4	HTTP .....	477
15.4.1	Verbindung mit GET.....	478
15.4.2	Verbindung mit POST .....	483
15.5	Mini-Webserver .....	483

### **16 Datenbankanbindung .....493**

16.1	C++-Interface.....	494
16.2	Anwendungsbeispiel.....	498

**Teil III: Praktische Methoden und Werkzeuge der Softwareentwicklung .....505****17 Abläufe automatisieren mit make .....507**

17.1 Quellen ..... 508

17.2 Wirkungsweise ..... 509

17.3 Variablen und Muster ..... 511

17.4 Universelles Makefile für einfache Projekte ..... 512

**18 Unit-Test ..... 515**

18.1 Werkzeuge ..... 516

18.2 Test Driven Development ..... 517

18.3 Boost Unit Test Framework ..... 518

18.3.1 Testaufbau ..... 519

18.3.2 Beispiel: Testgetriebene Entwicklung einer Operatorfunktion ..... 520

18.3.3 Fixture ..... 524

18.3.4 Testprotokoll und Log-Level ..... 525

18.3.5 Prüf-Makros ..... 526

18.3.6 Kommandozeilen-Optionen ..... 530

**19 Werkzeuge zur Verwaltung von Projekten ..... 531**

19.1 Dokumentation und Strukturanalyse mit doxygen ..... 531

19.1.1 Strukturanalyse ..... 535

19.2 Versionskontrolle ..... 536

19.2.1 Einrichtung des Servers ..... 538

19.2.2 Exemplarische Benutzung ..... 541

19.3 Projektverwaltung ..... 543

19.3.1 Projektmanagement ..... 543

19.3.2 Wiki für Software-Entwicklungsprojekte ..... 543

**Teil IV: Das C++-Rezeptbuch: Tipps und Lösungen für typische Aufgaben.....545****20 Sichere Programmentwicklung .....547**

20.1 Regeln zum Design von Methoden ..... 547

20.2 Defensive Programmierung ..... 550

20.2.1 double- und float-Werte richtig vergleichen ..... 551

20.2.2 const verwenden ..... 551

20.2.3 Anweisungen nach for/if/while einklammern ..... 552

20.2.4 int und unsigned nicht mischen ..... 552



20.2.5	Postfix++ mit Präfix++ implementieren .....	552
20.2.6	Ein Destruktor darf keine Exception werfen .....	553
20.2.7	Typumwandlungsoperatoren vermeiden .....	554
20.2.8	explicit-Konstruktoren bevorzugen .....	554
20.2.9	Leere Standardkonstruktoren vermeiden .....	554
20.2.10	Kopieren und Zuweisung verbieten .....	554
20.2.11	Vererbung verbieten .....	556
20.2.12	Defensiv Objekte löschen .....	557
20.3	Exception-sichere Beschaffung von Ressourcen .....	557
20.3.1	Sichere Verwendung von shared_ptr .....	557
20.3.2	shared_ptr für Arrays korrekt verwenden .....	558
20.3.3	Exception-sichere Funktion .....	558
20.3.4	Exception-sicherer Konstruktor .....	559
20.3.5	Exception-sichere Zuweisung .....	560
20.4	Aussagefähige Fehlermeldung ohne neuen String erzeugen .....	562
20.5	Empfehlungen zur Thread-Programmierung .....	563
20.5.1	Warten auf die Freigabe von Ressourcen .....	563
20.5.2	Deadlock-Vermeidung .....	564
20.5.3	notify_all oder notify_one? .....	564
20.5.4	Performance mit Threads verbessern? .....	565
<b>21</b>	<b>Von der UML nach C++ .....</b>	<b>567</b>
21.1	Vererbung .....	568
21.2	Interface anbieten und nutzen .....	568
21.3	Assoziation .....	570
21.3.1	Aggregation .....	573
21.3.2	Komposition .....	573
<b>22</b>	<b>Performance, Wert- und Referenzsemantik .....</b>	<b>575</b>
22.1	Performanceproblem Wertsemantik .....	577
22.2	Optimierung durch Referenzsemantik für R-Werte .....	578
22.3	Ein effizienter binärer Plusoperator .....	580
<b>23</b>	<b>Effektive Programmerzeugung .....</b>	<b>585</b>
23.1	Automatische Ermittlung von Abhängigkeiten .....	586
23.1.1	Getrennte Verzeichnisse: src, obj, bin .....	587
23.2	Makefile für Verzeichnisbäume .....	589
23.2.1	Rekursive Make-Aufrufe .....	590

23.2.2	Ein Makefile für alles .....	592
23.3	Automatische Erzeugung von Makefiles.....	593
23.3.1	Makefile für rekursive Aufrufe erzeugen .....	594
23.4	Erzeugen von Bibliotheken .....	595
23.4.1	Statische Bibliotheksmodule.....	596
23.4.2	Dynamische Bibliotheksmodule.....	597
23.5	GNU Autotools .....	600
23.6	CMake .....	603
23.7	Code Bloat bei der Instanziierung von Templates vermeiden.....	603
23.7.1	extern-Template.....	605
23.7.2	Aufspaltung in Schnittstelle und Implementation.....	606
23.7.3	export-Template.....	607
<b>24</b>	<b>Algorithmen für verschiedene Aufgaben.....</b>	<b>609</b>
24.1	Algorithmen mit Strings .....	610
24.1.1	String splitten .....	610
24.1.2	String in Zahl umwandeln .....	611
24.1.3	Zahl in String umwandeln .....	612
24.1.4	Strings sprachlich richtig sortieren .....	613
24.1.5	Umwandlung in Klein- bzw. Großschreibung.....	614
24.1.6	Strings sprachlich richtig vergleichen.....	616
24.1.7	Von der Groß-/Kleinschreibung unabhängiger Zeichenvergleich .....	617
24.1.8	Von der Groß-/Kleinschreibung unabhängige Suche .....	618
24.2	Textverarbeitung.....	619
24.2.1	Datei durchsuchen .....	619
24.2.2	Ersetzungen in einer Datei.....	621
24.2.3	Code-Formatierer .....	623
24.2.4	Lines of Code (LOC) ermitteln .....	624
24.2.5	Zeilen, Wörter und Zeichen einer Datei zählen .....	626
24.2.6	CSV-Datei lesen .....	626
24.2.7	Kreuzreferenzliste .....	627
24.3	Operationen auf Folgen .....	630
24.3.1	Container anzeigen .....	630
24.3.2	Folge mit gleichen Werten initialisieren .....	631
24.3.3	Folge mit Werten eines Generators initialisieren .....	631
24.3.4	Folge mit fortlaufenden Werten initialisieren.....	632
24.3.5	Summe und Produkt.....	632

24.3.6	Mittelwert und Standardabweichung .....	633
24.3.7	Skalarprodukt.....	634
24.3.8	Folge der Teilsummen oder -produkte .....	635
24.3.9	Folge der Differenzen.....	636
24.3.10	Minimum und Maximum .....	637
24.3.11	Elemente rotieren .....	639
24.3.12	Elemente verwürfeln.....	640
24.3.13	Dubletten entfernen .....	641
24.3.14	Reihenfolge umdrehen .....	643
24.3.15	Anzahl der Elemente, die einer Bedingung genügen.....	644
24.3.16	Gilt X für alle, keins oder wenigstens ein Element einer Folge? .....	645
24.3.17	Permutationen .....	646
24.3.18	Lexikografischer Vergleich.....	647
24.4	Sortieren und Verwandtes .....	648
24.4.1	Partitionieren .....	648
24.4.2	Sortieren.....	650
24.4.3	Stabiles Sortieren .....	650
24.4.4	Partielles Sortieren .....	651
24.4.5	Das n.-größte oder n.-kleinste Element finden .....	652
24.4.6	Verschmelzen (merge) .....	653
24.5	Suchen und Finden .....	657
24.5.1	Element finden .....	657
24.5.2	Element einer Menge in der Folge finden .....	658
24.5.3	Teilfolge finden.....	659
24.5.4	Bestimmte benachbarte Elemente finden .....	661
24.5.5	Bestimmte aufeinanderfolgende Werte finden .....	662
24.5.6	Binäre Suche.....	664
24.6	Mengenoperationen auf sortierten Strukturen .....	667
24.6.1	Teilmengenrelation .....	667
24.6.2	Vereinigung.....	668
24.6.3	Schnittmenge .....	669
24.6.4	Differenz .....	669
24.6.5	Symmetrische Differenz .....	670
24.7	Heap-Algorithmen .....	671
24.7.1	pop_heap.....	672
24.7.2	push_heap.....	673
24.7.3	make_heap .....	674

24.7.4	sort_heap .....	674
24.7.5	is_heap .....	675
24.8	Vergleich von Containern auch ungleichen Typs.....	675
24.8.1	Unterschiedliche Elemente finden .....	675
24.8.2	Prüfung auf gleiche Inhalte .....	677
24.9	Rechnen mit komplexen Zahlen: Der C++-Standardtyp complex.....	678
24.10	Schnelle zweidimensionale Matrix mit zusammenhängendem Speicher.....	680
24.10.1	Optimierung mathematischer Array-Operationen .....	685
24.11	Vermischtes .....	689
24.11.1	Erkennung eines Datums.....	689
24.11.2	Erkennung einer IP-Adresse .....	691
24.11.3	Erzeugen von Zufallszahlen .....	692
24.11.4	for_each – Auf jedem Element eine Funktion ausführen .....	693
24.11.5	Verschiedene Möglichkeiten, Container-Bereiche zu kopieren.....	693
24.11.6	Vertauschen von Elementen, Bereichen und Containern .....	696
24.11.7	Elemente transformieren .....	696
24.11.8	Ersetzen und Varianten .....	698
24.11.9	Elemente herausfiltern .....	700
24.11.10	Minimum und Maximum .....	701
24.11.11	Grenzwerte von Zahltypen .....	701
<b>25</b>	<b>Ein- und Ausgabe .....</b>	<b>703</b>
25.1	Datei- und Verzeichnisoperationen.....	703
25.1.1	Datei oder Verzeichnis löschen.....	704
25.1.2	Datei oder Verzeichnis umbenennen .....	705
25.1.3	Verzeichnis anlegen.....	706
25.1.4	Verzeichnis anzeigen .....	707
25.1.5	Verzeichnisbaum anzeigen.....	708
25.2	Tabelle formatiert ausgeben .....	710
25.3	Formatierte Daten lesen.....	711
25.3.1	Eingabe benutzerdefinierter Typen .....	711
25.4	Array als Block lesen oder schreiben.....	713
<b>Teil V:</b>	<b>Die C++-Standardbibliothek .....</b>	<b>715</b>
<b>26</b>	<b>Aufbau und Übersicht .....</b>	<b>717</b>
26.1	Auslassungen.....	719

26.2	Beispiele des Buchs und die C++-Standardbibliothek .....	721
<b>27</b>	<b>Hilfsfunktionen und -klassen .....</b>	<b>723</b>
27.1	Relationale Operatoren .....	723
27.2	Unterstützung der Referenzsemantik für R-Werte .....	724
27.3	Paare .....	727
27.4	Tupel .....	729
27.5	Funktionsobjekte .....	729
27.5.1	Arithmetische, vergleichende und logische Operationen .....	730
27.5.2	Funktionsobjekte zum Negieren logischer Prädikate .....	731
27.5.3	Binden von Argumentwerten .....	732
27.5.4	Zeiger auf Funktionen in Objekte umwandeln .....	733
27.6	Templates für rationale Zahlen .....	734
27.7	Zeit und Dauer .....	735
27.8	Hüllklasse für Referenzen .....	735
<b>28</b>	<b>Container .....</b>	<b>737</b>
28.1	Gemeinsame Eigenschaften .....	739
28.1.1	Initialisierungslisten .....	741
28.1.2	Konstruktion an Ort und Stelle .....	742
28.1.3	Reversible Container .....	742
28.2	Sequenzen .....	743
28.2.1	vector .....	744
28.2.2	vector<bool> .....	745
28.2.3	list .....	746
28.2.4	deque .....	748
28.2.5	stack .....	749
28.2.6	queue .....	751
28.2.7	priority_queue .....	752
28.2.8	array .....	754
28.3	Sortierte assoziative Container .....	756
28.3.1	map .....	756
28.3.2	multimap .....	760
28.3.3	set .....	761
28.3.4	multiset .....	764
28.4	Hash-Container .....	765
28.4.1	unordered_map .....	767
28.4.2	unordered_multimap .....	771

28.4.3	<code>unordered_set</code> .....	772
28.4.4	<code>unordered_multiset</code> .....	774
28.5	<code>bitset</code> .....	775
<b>29</b>	<b>Iteratoren</b> .....	<b>779</b>
29.1	Iterator-Kategorien .....	780
29.1.1	Anwendung von Traits .....	782
29.2	<code>distance()</code> und <code>advance()</code> .....	784
29.3	Reverse-Iteratoren .....	785
29.4	Insert-Iteratoren .....	786
29.5	Stream-Iteratoren .....	787
<b>30</b>	<b>Algorithmen</b> .....	<b>789</b>
30.1	Algorithmen mit Prädikat .....	790
30.1.1	Algorithmen mit binärem Prädikat .....	790
30.2	Übersicht .....	791
<b>31</b>	<b>Nationale Besonderheiten</b> .....	<b>795</b>
31.1	Sprachumgebungen festlegen und ändern .....	796
31.1.1	Die locale-Funktionen .....	797
31.2	Zeichensätze und -codierung .....	798
31.3	Zeichenklassifizierung und -umwandlung .....	802
31.4	Kategorien .....	803
31.4.1	<code>collate</code> .....	803
31.4.2	<code>ctype</code> .....	804
31.4.3	<code>numeric</code> .....	805
31.4.4	<code>monetary</code> .....	807
31.4.5	<code>time</code> .....	810
31.4.6	<code>messages</code> .....	812
31.5	Konstruktion eigener Facetten .....	813
<b>32</b>	<b>String</b> .....	<b>815</b>
<b>33</b>	<b>Speichermanagement</b> .....	<b>823</b>
33.1	Smart Pointer <code>unique_ptr</code> , <code>shared_ptr</code> , <code>weak_ptr</code> .....	823
33.2	<code>new</code> mit vorgegebenem Speicherort .....	828
33.3	Hilfsfunktionen .....	829

<b>34</b>	<b>Optimierte numerische Arrays (valarray)</b>	<b>831</b>
34.1	Konstruktoren	832
34.2	Elementfunktionen	832
34.3	Binäre Valarray-Operatoren	835
34.4	Mathematische Funktionen	837
34.5	slice und slice_array	838
34.6	gslice und gslice_array	840
34.7	mask_array	843
34.8	indirect_array	844
<b>35</b>	<b>C-Header</b>	<b>847</b>
35.1	<cassert>	848
35.2	<cctype>	848
35.3	<cerrno>	849
35.4	<cmath>	849
35.5	<cstdarg>	850
35.6	<cstddef>	851
35.7	<cstdio>	851
35.8	<cstdlib>	851
35.9	<cstring>	853
35.10	<ctime>	855
<b>A</b>	<b>Anhang</b>	<b>857</b>
A.1	Programmierhinweise	857
A.2	C++-Schlüsselwörter	860
A.3	ASCII-Tabelle	861
A.4	Rangfolge der Operatoren	862
A.5	Compilerbefehle	863
A.6	Änderungen des C++-Standards	864
A.6.1	Änderungen/Erweiterungen der C++-Sprache	864
A.6.2	Änderungen/Erweiterungen der C++-Standardbibliothek	871
A.7	Lösungen zu den Übungsaufgaben	872
A.8	Installation der DVD-Software für Windows	917
A.8.1	Installation des Compilers und einiger Bibliotheken	917
A.8.2	Bei Verzicht auf die automatische Installation	918
A.8.3	Codeblocks einrichten	919
A.8.4	Integration von Qt in ein Code::Blocks-Projekt	921
A.9	Installation der DVD-Software für Linux	922

A.9.1	Installation des Compilers .....	922
A.9.2	Installation von Boost .....	923
A.9.3	Installation von Code::Blocks .....	923
A.9.4	Code::Blocks einrichten .....	924
A.9.5	Beispieldateien entpacken .....	924
A.9.6	Installation von Qt4 .....	925
A.9.7	Integration von Qt in ein Code::Blocks-Projekt.....	926
<b>Literaturverzeichnis</b>		<b>927</b>
<b>Glossar</b>		<b>931</b>
<b>Register</b>		<b>941</b>