

Inhalt

Vorwort	XXXI
Teil I – Einführung	1
1 Vorbemerkungen	3
1.1 Zum Aufbau dieses Buchs	3
1.1.1 Einführung	4
1.1.2 Grundlegende Sprachmittel	4
1.1.3 Abstraktionsmechanismen	5
1.1.4 Die Standardbibliothek	7
1.1.5 Beispiele und Referenzen	8
1.2 Der Entwurf von C++	10
1.2.1 Programmierstile	12
1.2.2 Typprüfung	15
1.2.3 C-Kompatibilität	16
1.2.4 Sprache, Bibliotheken und Systeme	17
1.3 C++ lernen	19
1.3.1 In C++ programmieren	21
1.3.2 Ratschläge für C++-Programmierer	22
1.3.3 Ratschläge für C-Programmierer	23
1.3.4 Ratschläge für Java-Programmierer	24
1.4 Historische Anmerkungen	25
1.4.1 Chronik	26
1.4.2 Die frühen Jahre	27
1.4.2.1 Sprachfeatures und Bibliotheksinstrumente	28
1.4.3 Der 1998-Standard	30
1.4.3.1 Sprachfeatures	30
1.4.3.2 Die Standardbibliothek	31
1.4.4 Der 2011-Standard	32
1.4.4.1 Sprachfeatures	33
1.4.4.2 Standardbibliothek	34
1.4.5 Wofür wird C++ verwendet?	35

1.5	Ratschläge	37
1.6	Literaturhinweise	38
2	Ein Rundreise durch C++: Die Grundlagen	43
2.1	Einführung	43
2.2	Die Grundlagen	44
2.2.1	Hello, World!	44
2.2.2	Typen, Variablen und Arithmetik	46
2.2.3	Konstanten	48
2.2.4	Tests und Schleifen	49
2.2.5	Zeiger, Arrays und Schleifen	51
2.3	Benutzerdefinierte Typen	53
2.3.1	Strukturen	53
2.3.2	Klassen	55
2.3.3	Aufzählungen	57
2.4	Modularität	58
2.4.1	Separate Kompilierung	59
2.4.2	Namespaces	60
2.4.3	Fehlerbehandlung	61
2.4.3.1	Ausnahmen	62
2.4.3.2	Invarianten	63
2.4.3.3	Statische Assertionen	64
2.5	Nachbemerkung	65
2.6	Ratschläge	65
3	Eine Rundreise durch C++: Abstraktionsmechanismen	67
3.1	Einführung	67
3.2	Klassen	68
3.2.1	Konkrete Typen	68
3.2.1.1	Ein arithmetischer Typ	69
3.2.1.2	Ein Container	71
3.2.1.3	Container initialisieren	72
3.2.2	Abstrakte Typen	73
3.2.3	Virtuelle Funktionen	76
3.2.4	Klassenhierarchien	77
3.3	Kopieren und verschieben	81
3.3.1	Container kopieren	81
3.3.2	Container verschieben	83
3.3.3	Ressourcenverwaltung	85
3.3.4	Operationen unterdrücken	86
3.4	Templates	87
3.4.1	Parametrisierte Typen	87
3.4.2	Funktions-Templates	88

3.4.3	Funktionsobjekte	89
3.4.4	Variadische Templates	92
3.4.5	Alias	93
3.5	Ratschläge	94
4	Eine Rundreise durch C++: Container und Algorithmen	95
4.1	Bibliotheken	95
4.1.1	Überblick über die Standardbibliothek	96
4.1.2	Header und Namespace der Standardbibliothek	97
4.2	Strings	98
4.3	Stream-Ein-/Ausgabe	100
4.3.1	Ausgabe	100
4.3.2	Eingabe	101
4.3.3	Ein-/Ausgabe von benutzerdefinierten Typen	102
4.4	Container	104
4.4.1	vector	104
4.4.1.1	Elemente	106
4.4.1.2	Bereichsüberprüfung	106
4.4.2	list	107
4.4.3	map	109
4.4.4	unordered_map	110
4.4.5	Überblick über Container	110
4.5	Algorithmen	112
4.5.1	Iteratoren verwenden	113
4.5.2	Iteratortypen	115
4.5.3	Stream-Iteratoren	116
4.5.4	Prädikate	118
4.5.5	Überblick über Algorithmen	118
4.5.6	Containeralgorithmen	119
4.6	Ratschläge	120
5	Eine Rundreise durch C++: Nebenläufigkeit und Dienstprogramme	121
5.1	Einführung	121
5.2	Ressourcenverwaltung	121
5.2.1	unique_ptr und shared_ptr	122
5.3	Nebenläufigkeit	124
5.3.1	Tasks und Threads	125
5.3.2	Argumente übergeben	126
5.3.3	Ergebnisse zurückgeben	127
5.3.4	Daten gemeinsam nutzen	127
5.3.4.1	Warten auf Ereignisse	129
5.3.5	Kommunizierende Tasks	130

5.3.5.1	future und promise	131
5.3.5.2	packaged_task	132
5.3.5.3	async()	133
5.4	Kleine Hilfskomponenten	134
5.4.1	Zeit	134
5.4.2	Typfunktionen	135
5.4.2.1	iterator_traits	135
5.4.2.2	Typprädikate	137
5.4.3	pair und tuple	138
5.5	Reguläre Ausdrücke	139
5.6	Mathematik	140
5.6.1	Mathematische Funktionen und Algorithmen	140
5.6.2	Komplexe Zahlen	140
5.6.3	Zufallszahlen	141
5.6.4	Vektorarithmetik	143
5.6.5	Numerische Grenzen	144
5.7	Ratschläge	144
Teil II – Grundlegende Sprachelemente		145
6	Typen und Deklarationen	147
6.1	Der ISO-C++-Standard	147
6.1.1	Implementierungen	149
6.1.2	Der grundlegende Quellzeichensatz	149
6.2	Typen	150
6.2.1	Fundamentale Typen	150
6.2.2	Boolesche Typen	151
6.2.3	Zeichentypen	153
6.2.3.1	Vorzeichenbehaftete und vorzeichenlose Zeichen	155
6.2.3.2	Zeichenlitterale	156
6.2.4	Ganzzahltypen	158
6.2.4.1	Ganzzahliliterale	158
6.2.4.2	Typen von Ganzzahliliteralen	159
6.2.5	Gleitkommatypen	160
6.2.5.1	Gleitkommaliterale	160
6.2.6	Präfixe und Suffixe	161
6.2.7	void	162
6.2.8	Größen	162
6.2.9	Ausrichtung	165
6.3	Deklarationen	166
6.3.1	Die Struktur von Deklarationen	168
6.3.2	Mehrere Namen deklarieren	169
6.3.3	Namen	170
6.3.3.1	Schlüsselwörter	171

6.3.4	Gültigkeitsbereiche	172
6.3.5	Initialisierung	174
6.3.5.1	Fehlende Initialisierer	177
6.3.5.2	Initialisierungslisten	178
6.3.6	Einen Typ herleiten: auto und decltype()	179
6.3.6.1	Der Typspezifizierer auto	179
6.3.6.2	auto und {} -Listen	180
6.3.6.3	Der Spezifizierer decltype()	181
6.4	Objekte und Werte	182
6.4.1	L-Werte und R-Werte	182
6.4.2	Lebensdauer von Objekten	183
6.5	Typalias	184
6.6	Ratschläge	185
7	Zeiger, Arrays und Referenzen	187
7.1	Einführung	187
7.2	Zeiger	187
7.2.1	void*	188
7.2.2	nullptr	189
7.3	Arrays	190
7.3.1	Array-Initialisierer	191
7.3.2	Stringlitterale	192
7.3.2.1	Rohe Zeichen-Strings	194
7.3.2.2	Größere Zeichensätze	195
7.4	Zeiger auf Arrays	196
7.4.1	Navigieren in Arrays	198
7.4.2	Mehrdimensionale Arrays	200
7.4.3	Arrays übergeben	201
7.5	Zeiger und const	203
7.6	Zeiger und Besitz	205
7.7	Referenzen	206
7.7.1	L-Wert-Referenzen	208
7.7.2	R-Wert-Referenzen	211
7.7.3	Referenzen auf Referenzen	214
7.7.4	Zeiger und Referenzen	215
7.8	Ratschläge	217
8	Strukturen, Unions und Aufzählungen	219
8.1	Einführung	219
8.2	Strukturen	219
8.2.1	Layout einer Struktur	221
8.2.2	Namen von Strukturen	222
8.2.3	Strukturen und Klassen	224

8.2.4	Strukturen und Arrays	225
8.2.5	Typäquivalenz	227
8.2.6	Plain Old Data	228
8.2.7	Felder	230
8.3	Unions	231
8.3.1	Unions und Klassen	233
8.3.2	Anonyme Unions	234
8.4	Aufzählungen	237
8.4.1	Aufzählungsklassen	237
8.4.2	Einfache Aufzählungen	241
8.4.3	Unbenannte Aufzählungen	243
8.5	Ratschläge	243
9	Anweisungen	245
9.1	Einführung	245
9.2	Zusammenfassung der Anweisungen	245
9.3	Deklarationen als Anweisungen	247
9.4	Auswahanweisungen	248
9.4.1	if -Anweisungen	248
9.4.2	switch -Anweisungen	250
9.4.2.1	Deklarationen in case -Zweigen	252
9.4.3	Deklarationen in Bedingungen	252
9.5	Schleifenanweisungen	253
9.5.1	Bereichsbasierte for -Anweisungen	254
9.5.2	for -Anweisungen	255
9.5.3	while -Anweisungen	256
9.5.4	do -Anweisungen	257
9.5.5	Schleifen verlassen	257
9.6	goto -Anweisungen	258
9.7	Kommentare und Einrückungen	259
9.8	Ratschläge	261
10	Ausdrücke	263
10.1	Einführung	263
10.2	Ein Taschenrechner	263
10.2.1	Der Parser	264
10.2.2	Eingabe	268
10.2.3	Low-level-Eingabe	272
10.2.4	Fehlerbehandlung	274
10.2.5	Das Rahmenprogramm	274
10.2.6	Header	275
10.2.7	Befehlszeilenargumente	276
10.2.8	Eine Anmerkung zum Stil	277

10.3	Zusammenfassung der Operatoren	278
10.3.1	Ergebnisse	282
10.3.2	Reihenfolge der Auswertung	283
10.3.3	Operatorrangfolge	284
10.3.4	Temporäre Objekte	285
10.4	Konstante Ausdrücke	287
10.4.1	Symbolische Konstanten	289
10.4.2	const -Typen in konstanten Ausdrücken	290
10.4.3	Literale Typen	290
10.4.4	Referenzargumente	291
10.4.5	Adresse konstanter Ausdrücke	292
10.5	Implizite Typkonvertierung	292
10.5.1	Heraufstufungen	293
10.5.2	Konvertierungen	293
10.5.2.1	Integrale Konvertierungen	294
10.5.2.2	Gleitkommakonvertierungen	294
10.5.2.3	Zeiger- und Referenzkonvertierungen	295
10.5.2.4	Zeiger-auf-Member-Konvertierungen	295
10.5.2.5	Boolesche Konvertierungen	295
10.5.2.6	Gleitkomma-Ganzzahl-Konvertierungen	296
10.5.3	Übliche arithmetische Konvertierungen	297
10.6	Ratschläge	297
11	Auswahloperationen	299
11.1	Diverse Operatoren	299
11.1.1	Logische Operatoren	299
11.1.2	Bitweise logische Operatoren	299
11.1.3	Bedingte Ausdrücke	301
11.1.4	Inkrementieren und Dekrementieren	301
11.2	Freispeicher	303
11.2.1	Speicherverwaltung	305
11.2.2	Arrays	308
11.2.3	Speicherplatz beschaffen	309
11.2.4	Überladen von new	310
11.2.4.1	nothrow new	312
11.3	Listen	313
11.3.1	Implementierungsmodell	313
11.3.2	Qualifizierte Listen	315
11.3.3	Unqualifizierte Listen	315
11.4	Lambda-Ausdrücke	317
11.4.1	Implementierungsmodelle	318
11.4.2	Alternativen für Lambda-Ausdrücke	319
11.4.3	Erfassung	321
11.4.3.1	Lambda und Lebensdauer	323

11.4.3.2	Namen von Namespaces	324
11.4.3.3	Lambda und this	324
11.4.3.4	Veränderbare Lambda-Ausdrücke	324
11.4.4	Aufruf und Rückgabe	325
11.4.5	Der Typ eines Lambda-Ausdrucks	325
11.5	Explizite Typumwandlung	326
11.5.1	Konstruktion	328
11.5.2	Benannte Typumwandlungen	329
11.5.3	C-Typumwandlungen	331
11.5.4	Funktionale Typumwandlung	331
11.6	Ratschläge	332
12	Funktionen	333
12.1	Funktionsdeklarationen	333
12.1.1	Warum Funktionen?	334
12.1.2	Bestandteile einer Funktionsdeklaration	334
12.1.3	Funktionsdefinitionen	335
12.1.4	Werte zurückgeben	337
12.1.5	Inline-Funktionen	339
12.1.6	constexpr -Funktionen	340
12.1.6.1	constexpr und Referenzen	341
12.1.6.2	Bedingte Auswertung	342
12.1.7	[[noreturn]] -Funktionen	342
12.1.8	Lokale Variablen	343
12.2	Argumentübergabe	344
12.2.1	Referenzargumente	345
12.2.2	Array-Argumente	347
12.2.3	Listenargumente	349
12.2.4	Nicht angegebene Anzahl von Argumenten	350
12.2.5	Standardargumente	354
12.3	Überladene Funktionen	356
12.3.1	Automatische Auflösung von Überladungen	356
12.3.2	Überladen und Rückgabetyt	358
12.3.3	Überladen und Gültigkeitsbereiche	359
12.3.4	Auflösung für mehrere Argumente	360
12.3.5	Manuelle Auflösung von Überladungen	360
12.4	Vor- und Nachbedingungen	361
12.5	Zeiger auf Funktion	363
12.6	Makros	367
12.6.1	Bedingte Übersetzung	370
12.6.2	Vordefinierte Makros	371
12.6.3	Pragmas	372
12.7	Ratschläge	372

13	Ausnahmenbehandlung	375
13.1	Fehlerbehandlung	375
13.1.1	Ausnahmen	376
13.1.2	Herkömmliche Fehlerbehandlung	378
13.1.3	Durchhangeln	379
13.1.4	Alternative Ansichten von Ausnahmen	380
13.1.4.1	Asynchrone Ereignisse	380
13.1.4.2	Ausnahmen, die keine Fehler sind	380
13.1.5	Wann Sie keine Ausnahmen verwenden können	381
13.1.6	Hierarchische Fehlerbehandlung	382
13.1.7	Ausnahmen und Effizienz	384
13.2	Ausnahmegarantien	386
13.3	Ressourcenverwaltung	388
13.3.1	finally	391
13.4	Invarianten erzwingen	393
13.5	Ausnahmen auslösen und abfangen	398
13.5.1	Ausnahmen auslösen	398
13.5.1.1	noexcept -Funktionen	400
13.5.1.2	Der Operator noexcept	400
13.5.1.3	Ausnahmespezifikationen	401
13.5.2	Ausnahmen abfangen	402
13.5.2.1	Ausnahmen erneut auslösen	403
13.5.2.2	Jede Ausnahme abfangen	404
13.5.2.3	Mehrere Handler	405
13.5.2.4	try -Blöcke in Funktionen	405
13.5.2.5	Beendigung	407
13.5.3	Ausnahmen und Threads	409
13.6	Eine vector -Implementierung	410
13.6.1	Ein einfacher vector	410
13.6.2	Speicher explizit darstellen	414
13.6.3	Zuweisung	417
13.6.4	Größe ändern	419
13.6.4.1	reserve()	420
13.6.4.2	resize()	421
13.6.4.3	push_back()	421
13.6.4.4	Abschließende Gedanken	422
13.7	Ratschläge	423
14	Namespaces	425
14.1	Kompositionsprobleme	425
14.2	Namespaces	426
14.2.1	Explizite Qualifizierung	428
14.2.2	using -Deklarationen	429
14.2.3	using -Direktiven	430

14.2.4	Argumentabhängige Namensauflösung	431
14.2.5	Namespaces sind offen	434
14.3	Modularisierung und Schnittstellen	435
14.3.1	Namespaces als Module	436
14.3.2	Implementierungen	438
14.3.3	Schnittstellen und Implementierungen	440
14.4	Komposition mit Namespaces	442
14.4.1	Komfort vs. Sicherheit	442
14.4.2	Namespace-Alias	443
14.4.3	Namespaces zusammensetzen	444
14.4.4	Komposition und Auswahl	445
14.4.5	Namespaces und Überladen	446
14.4.6	Versionsverwaltung	448
14.4.7	Verschachtelte Namespaces	450
14.4.8	Unbenannte Namespaces	451
14.4.9	C-Header	452
14.5	Ratschläge	453
15	Quelldateien und Programme	455
15.1	Separate Übersetzung	455
15.2	Binden	456
15.2.1	Dateilokale Namen	459
15.2.2	Header-Dateien	460
15.2.3	Die Eine-Definition-Regel	462
15.2.4	Header der Standardbibliothek	464
15.2.5	Binden mit Nicht-C++-Code	465
15.2.6	Binden und Zeiger auf Funktionen	467
15.3	Header-Dateien verwenden	468
15.3.1	Organisation mit einzeltem Header	468
15.3.2	Organisation mit mehreren Header-Dateien	472
15.3.2.1	Andere Taschenrechnermodule	475
15.3.2.2	Header verwenden	477
15.3.3	Include-Wächter	478
15.4	Programme	479
15.4.1	Initialisierung von nichtlokalen Variablen	479
15.4.2	Initialisierung und Nebenläufigkeit	480
15.4.3	Programmbeendigung	481
15.5	Ratschläge	483
Teil III	– Abstraktionsmechanismen	485
16	Klassen	487
16.1	Einführung	487
16.2	Grundlagen von Klassen	488

16.2.1	Member-Funktionen	489
16.2.2	Standardmäßiges Kopieren	490
16.2.3	Zugriffskontrolle	491
16.2.4	class und struct	492
16.2.5	Konstruktoren	494
16.2.6	Explizite Konstruktoren	496
16.2.7	Klasseninterne Initialisierer	498
16.2.8	Klasseninterne Funktionsdefinitionen	499
16.2.9	Veränderlichkeit	500
16.2.9.1	Konstante Member-Funktionen	500
16.2.9.2	Physische und logische Konstanz	501
16.2.9.3	mutable	502
16.2.9.4	Veränderlichkeit über Indirektion	502
16.2.10	Selbstreferenz	503
16.2.11	Member-Zugriff	505
16.2.12	Statische Member	506
16.2.13	Member-Typen	508
16.3	Konkrete Klassen	509
16.3.1	Member-Funktionen	512
16.3.2	Hilfsfunktionen	515
16.3.3	Überladene Operatoren	516
16.3.4	Der Stellenwert von konkreten Klassen	517
16.4	Ratschläge	518
17	Konstruieren, Aufräumen, Kopieren und Verschieben	521
17.1	Einführung	521
17.2	Konstruktoren und Destruktoren	523
17.2.1	Konstruktoren und Invarianten	524
17.2.2	Destruktoren und Ressourcen	525
17.2.3	Basis- und Member-Destruktoren	526
17.2.4	Konstruktoren und Destruktoren aufrufen	527
17.2.5	Virtuelle Destruktoren	528
17.3	Initialisierung von Klassenobjekten	529
17.3.1	Initialisierung ohne Konstruktoren	529
17.3.2	Initialisierung mithilfe von Konstruktoren	531
17.3.2.1	Initialisierung durch Konstruktoren	533
17.3.3	Standardkonstruktoren	534
17.3.4	Initialisierungslisten-Konstruktoren	536
17.3.4.1	Mehrdeutigkeiten bei Initialisierungslisten-Konstruktoren auflösen	537
17.3.4.2	Initialisierungslisten verwenden	538
17.3.4.3	Direkte und Kopierinitialisierung	539
17.4	Initialisierung von Mitgliedern und Basisklassen	541
17.4.1	Member-Initialisierung	541
17.4.1.1	Member-Initialisierung und -Zuweisung	542

17.4.2	Basisklassen-Initialisierer	543
17.4.3	Konstruktoren delegieren	543
17.4.4	Klasseninterne Initialisierer	544
17.4.5	Initialisierer statischer Member	546
17.5	Kopieren und verschieben	547
17.5.1	Kopieren	548
17.5.1.1	Vorsicht vor Standardkonstruktoren	550
17.5.1.2	Kopieren von Basisklassen	550
17.5.1.3	Was Kopieren bedeutet	551
17.5.1.4	Slicing	554
17.5.2	Verschieben	555
17.6	Standardoperationen generieren	559
17.6.1	Explizite Standardoperationen	560
17.6.2	Standardoperationen	561
17.6.3	Standardoperationen verwenden	562
17.6.3.1	Standardkonstruktoren	562
17.6.3.2	Invarianten bewahren	562
17.6.3.3	Ressourceninvarianten	563
17.6.3.4	Partiell spezifizierte Invarianten	564
17.6.4	Gelöschte Funktionen	566
17.7	Ratschläge	568
18	Überladen von Operatoren	571
18.1	Einführung	571
18.2	Operatorfunktionen	573
18.2.1	Binäre und unäre Operatoren	574
18.2.2	Vordefinierte Bedeutungen für Operatoren	575
18.2.3	Operatoren und benutzerdefinierte Typen	576
18.2.4	Objekte übergeben	576
18.2.5	Operatoren in Namespaces	578
18.3	Ein Typ für komplexe Zahlen	580
18.3.1	Member- und Nicht-Member-Operatoren	580
18.3.2	Arithmetik mit gemischten Datentypen	581
18.3.3	Konvertierungen	582
18.3.3.1	Konvertierung von Operanden	584
18.3.4	Literale	585
18.3.5	Zugriffsfunktionen	586
18.3.6	Hilfsfunktionen	587
18.4	Typumwandlung	588
18.4.1	Konvertierungsoperatoren	588
18.4.2	Explizite Konvertierungsoperatoren	590
18.4.3	Mehrdeutigkeiten	591
18.5	Ratschläge	593

19	Spezielle Operatoren	595
19.1	Einführung	595
19.2	Spezielle Operatoren	595
19.2.1	Indizierung	595
19.2.2	Funktionsaufruf	596
19.2.3	Dereferenzieren	598
19.2.4	Inkrementieren und Dekrementieren	600
19.2.5	Allokation und Deallokation	602
19.2.6	Benutzerdefinierte Literale	604
19.3	Eine String -Klasse	607
19.3.1	Wesentliche Operationen	608
19.3.2	Zugriff auf Zeichen	608
19.3.3	Darstellung	609
19.3.3.1	Ergänzende Funktionen	611
19.3.4	Member-Funktionen	612
19.3.5	Hilfsfunktionen	615
19.3.6	Unsere String -Klasse verwenden	617
19.4	Friends	617
19.4.1	Friends finden	619
19.4.2	Friends und Member	620
19.5	Ratschläge	622
20	Abgeleitete Klassen	625
20.1	Einführung	625
20.2	Abgeleitete Klassen	626
20.2.1	Member-Funktionen	629
20.2.2	Konstruktoren und Destruktoren	630
20.3	Klassenhierarchien	631
20.3.1	Typfelder	631
20.3.2	Virtuelle Funktionen	634
20.3.3	Explizite Qualifizierung	637
20.3.4	Überschreiben steuern	637
20.3.4.1	override	639
20.3.4.2	final	640
20.3.5	Member der Basisklasse verwenden	642
20.3.5.1	Konstruktoren vererben	643
20.3.6	Lockerung bei Rückgabetypen	645
20.4	Abstrakte Klassen	647
20.5	Zugriffskontrolle	649
20.5.1	Geschützte Member	653
20.5.1.1	Geschützte Member verwenden	654
20.5.2	Zugriff auf Basisklassen	654
20.5.2.1	Mehrfachvererbung und Zugriffskontrolle	655
20.5.3	using -Deklarationen und Zugriffskontrolle	656

20.6	Zeiger auf Member	657
20.6.1	Zeiger auf Funktions-Member	657
20.6.2	Zeiger auf Daten-Member	659
20.6.3	Basis- und abgeleitete Member	660
20.7	Ratschläge	661
21	Klassenhierarchien	663
21.1	Einführung	663
21.2	Klassenhierarchien entwerfen	663
21.2.1	Implementierungsvererbung	664
21.2.1.1	Kritische Betrachtungen	667
21.2.2	Schnittstellenvererbung	668
21.2.3	Alternative Implementierungen	670
21.2.3.1	Kritische Betrachtungen	673
21.2.4	Objekterstellung örtlich begrenzen	674
21.3	Mehrfachvererbung	675
21.3.1	Mehrfachschnittstellen	676
21.3.2	Mehrere Implementierungsklassen	676
21.3.3	Auflösung von Mehrdeutigkeiten	678
21.3.4	Eine Basisklasse wiederholt verwenden	682
21.3.5	Virtuelle Basisklassen	683
21.3.5.1	Virtuelle Basisklassen konstruieren	686
21.3.5.2	Member einer virtuellen Klasse nur einmal aufrufen	687
21.3.6	Replizierte und virtuelle Basisklassen	689
21.3.6.1	Funktionen virtueller Basisklassen überschreiben	691
21.4	Ratschläge	692
22	Laufzeit-Typinformationen	693
22.1	Einführung	693
22.2	Navigation in der Klassenhierarchie	693
22.2.1	dynamic_cast	695
22.2.1.1	dynamic_cast in Referenz	697
22.2.2	Mehrfachvererbung	698
22.2.3	static_cast und dynamic_cast	700
22.2.4	Eine Schnittstelle zurückgewinnen	701
22.3	Doppelte Bindung und Besucher	705
22.3.1	Doppelte Bindung	706
22.3.2	Besucher	708
22.4	Konstruktion und Destruktion	710
22.5	Typidentifizierung	711
22.5.1	Erweiterte Typinformationen	713
22.6	RTTI – richtig und falsch eingesetzt	714
22.7	Ratschläge	716

23	Templates	719
23.1	Einführung und Überblick	719
23.2	Ein einfaches String-Template	722
23.2.1	Ein Template definieren	724
23.2.2	Template-Instanziierung	725
23.3	Typprüfung	726
23.3.1	Typäquivalenz	728
23.3.2	Fehlererkennung	729
23.4	Member von Klassen-Templates	730
23.4.1	Daten-Member	730
23.4.2	Member-Funktionen	731
23.4.3	Member-Typalias	731
23.4.4	Statische Member	732
23.4.5	Member-Typen	732
23.4.6	Member-Templates	733
23.4.6.1	Templates und Konstruktoren	734
23.4.6.2	Templates und virtual	735
23.4.6.3	Verschachtelungen	735
23.4.7	Friends	738
23.5	Funktions-Templates	739
23.5.1	Argumente von Funktions-Templates	741
23.5.2	Funktions-Template-Argumente herleiten	742
23.5.2.1	Referenzherleitung	743
23.5.3	Überladen von Funktions-Templates	745
23.5.3.1	Mehrdeutigkeiten auflösen	746
23.5.3.2	Fehler bei Argumentersetzung	747
23.5.3.3	Überladen und Ableitung	749
23.5.3.4	Überladen und nicht hergeleitete Parameter	749
23.6	Template-Alias	750
23.7	Quellcodeorganisation	751
23.7.1	Binden	753
23.8	Ratschläge	754
24	Generische Programmierung	757
24.1	Einführung	757
24.2	Algorithmen und Lifting	758
24.3	Konzepte	762
24.3.1	Ein Konzept erkennen	763
24.3.2	Konzepte und Einschränkungen	766
24.4	Konzepte konkret machen	768
24.4.1	Axiome	772
24.4.2	Konzepte mit mehreren Argumenten	773
24.4.3	Wertkonzepte	775

24.4.4	Einschränkungsüberprüfungen	775
24.4.5	Überprüfung von Template-Definitionen	777
24.5	Ratschläge	779
25	Spezialisierung	781
25.1	Einführung	781
25.2	Template-Parameter und -Argumente	782
25.2.1	Typen als Argumente	782
25.2.2	Werte als Argumente	784
25.2.3	Operationen als Argumente	786
25.2.4	Templates als Argumente	788
25.2.5	Template-Standardargumente	789
25.2.5.1	Standardargumente bei Funktions-Templates	790
25.3	Spezialisierung	791
25.3.1	Schnittstellenspezialisierung	794
25.3.1.1	Implementierungsspezialisierung	795
25.3.2	Das primäre Template	795
25.3.3	Reihenfolge der Spezialisierung	797
25.3.4	Funktions-Template-Spezialisierung	798
25.3.4.1	Spezialisierung und Überladen	798
25.3.4.2	Spezialisierung, die kein Überladen ist	799
25.4	Ratschläge	800
26	Instanziierung	801
26.1	Einführung	801
26.2	Template-Instanziierung	802
26.2.1	Wann wird Instanziierung gebraucht?	803
26.2.2	Manuelle Kontrolle der Instanziierung	804
26.3	Namensbindung	805
26.3.1	Abhängige Namen	807
26.3.2	Bindung am Punkt der Definition	809
26.3.3	Bindung am Punkt der Instanziierung	810
26.3.4	Mehrere Punkte der Instanziierung	812
26.3.5	Templates und Namespaces	814
26.3.6	Zu aggressive ADL	815
26.3.7	Namen aus Basisklassen	817
26.4	Ratschläge	819
27	Templates und Hierarchien	821
27.1	Einführung	821
27.2	Parametrisierung und Hierarchie	822
27.2.1	Generierte Typen	824
27.2.2	Template-Konvertierungen	826

27.3	Hierarchien von Klassen-Templates	827
27.3.1	Templates als Schnittstellen	829
27.4	Template-Parameter als Basisklassen	829
27.4.1	Datenstrukturen zusammensetzen	830
27.4.2	Klassenhierarchien linearisieren	834
27.5	Ratschläge	839
28	Metaprogrammierung	841
28.1	Einführung	841
28.2	Typfunktionen	843
28.2.1	Typalias	846
28.2.1.1	Wann man keinen Alias verwendet	847
28.2.2	Typprädikate	848
28.2.3	Eine Funktion auswählen	850
28.2.4	Traits	850
28.3	Steuerungsstrukturen	852
28.3.1	Auswahl	852
28.3.1.1	Zwischen zwei Typen auswählen	853
28.3.1.2	Übersetzungszeit versus Laufzeit	853
28.3.1.3	Zwischen mehreren Typen auswählen	855
28.3.2	Iteration und Rekursion	856
28.3.2.1	Rekursion mit Klassen	857
28.3.3	Wann man Metaprogrammierung verwendet	857
28.4	Bedingte Definition: Enable_if	859
28.4.1	Enable_if verwenden	860
28.4.2	Enable_if implementieren	862
28.4.3	Enable_if und Konzepte	863
28.4.4	Weitere Beispiele mit Enable_if	863
28.5	Eine Liste zur Übersetzungszeit: Tuple	866
28.5.1	Eine einfache Ausgabefunktion	868
28.5.2	Elementzugriff	869
28.5.2.1	Konstante Tupel	871
28.5.3	make_tuple	872
28.6	Variadische Templates	873
28.6.1	Eine typsichere printf() -Funktion	873
28.6.2	Technische Details	876
28.6.3	Weiterleitung	878
28.6.4	Der Typ tuple der Standardbibliothek	879
28.7	Beispiel: SI-Einheiten	883
28.7.1	Einheiten	883
28.7.2	Größen	884
28.7.3	Literale für Einheiten	886
28.7.4	Hilfsfunktionen	888
28.8	Ratschläge	889

29	Ein Matrix-Design	891
29.1	Einführung	891
29.1.1	Einfache Anwendungen von Matrix	891
29.1.2	Anforderungen an Matrix	893
29.2	Ein Matrix -Template	894
29.2.1	Konstruktion und Zuweisung	896
29.2.2	Indizierung und Slicing	897
29.3	Arithmetische Matrix -Operationen	900
29.3.1	Skalaroperationen	901
29.3.2	Addition	901
29.3.3	Multiplikation	903
29.4	Matrix -Implementierung	905
29.4.1	slice()	905
29.4.2	Matrix -Slices	905
29.4.3	Matrix_ref	907
29.4.4	Matrix -Listeninitialisierung	908
29.4.5	Matrix -Zugriff	910
29.4.6	Nulldimensionale Matrix	913
29.5	Lineare Gleichungen lösen	914
29.5.1	Das klassische gaußsche Eliminationsverfahren	915
29.5.2	Pivotisierung	916
29.5.3	Testen	917
29.5.4	Verschmolzene Operationen	918
29.6	Ratschläge	921
Teil IV – Die Standardbibliothek		923
30	Überblick über die Standardbibliothek	925
30.1	Einführung	925
30.1.1	Komponenten der Standardbibliothek	926
30.1.2	Design einschränkungen	927
30.1.3	Beschreibungsstil	929
30.2	Header	929
30.3	Sprachunterstützung	934
30.3.1	Unterstützung für Initialisierungslisten	934
30.3.2	Unterstützung für bereichsbasierte for -Anweisung	935
30.4	Fehlerbehandlung	935
30.4.1	Ausnahmen	936
30.4.1.1	Die Hierarchie der Standardausnahmen	937
30.4.1.2	Weiterleitung von Ausnahmen	938
30.4.1.3	terminate()	941
30.4.2	Assertionen	941
30.4.3	system_error	942
30.4.3.1	Fehlercodes	943

30.4.3.2	Fehlerkategorien	945
30.4.3.3	system_error -Ausnahme	946
30.4.3.4	Potenziell portable Fehlerbedingungen	947
30.4.3.5	Fehlercodes zuordnen	947
30.4.3.6	errc -Fehlercodes	949
30.4.3.7	future_errc -Fehlercodes	952
30.4.3.8	io_errc -Fehlercodes	952
30.5	Ratschläge	952
31	STL-Container	955
31.1	Einführung	955
31.2	Überblick über Container	955
31.2.1	Containerdarstellung	958
31.2.2	Anforderungen an die Elemente	960
31.2.2.1	Vergleiche	961
31.2.2.2	Andere relationale Operatoren	962
31.3	Operatoren im Überblick	963
31.3.1	Member-Typen	966
31.3.2	Konstruktoren, Destruktor und Zuweisungen	967
31.3.3	Größe und Kapazität	969
31.3.4	Iteratoren	969
31.3.5	Elementzugriff	971
31.3.6	Stack-Operationen	971
31.3.7	Listenoperationen	972
31.3.8	Andere Operationen	973
31.4	Container	974
31.4.1	vector	974
31.4.1.1	vector und Wachstum	974
31.4.1.2	vector und Verschachtelung	976
31.4.1.3	vector und Arrays	978
31.4.1.4	vector und string	978
31.4.2	Listen	979
31.4.3	Assoziative Container	981
31.4.3.1	Geordnete assoziative Container	982
31.4.3.2	Ungeordnete assoziative Container	986
31.4.3.3	Ungeordnete Maps konstruieren	987
31.4.3.4	Hash- und Gleichheitsfunktionen	989
31.4.3.5	Lastfaktor und Buckets	992
31.5	Containeradapter	993
31.5.1	stack	994
31.5.2	queue	996
31.5.3	priority_queue	996
31.6	Ratschläge	997

32 STL-Algorithmen	1001
32.1 Einführung	1001
32.2 Algorithmen	1001
32.2.1 Sequenzen	1002
32.3 Richtlinienargumente	1003
32.3.1 Komplexität	1005
32.4 Nichtmodifizierende Sequenzalgorithmen	1006
32.4.1 for_each()	1006
32.4.2 Sequenzprädikate	1006
32.4.3 count()	1007
32.4.4 find()	1007
32.4.5 equal() und mismatch()	1008
32.4.6 search()	1009
32.5 Modifizierende Sequenzalgorithmen	1010
32.5.1 copy()	1011
32.5.2 unique()	1012
32.5.3 remove() , reverse() und replace()	1013
32.5.4 rotate() , random_shuffle() und partition()	1014
32.5.5 Permutationen	1015
32.5.6 fill()	1016
32.5.7 swap()	1017
32.6 Sortieren und Suchen	1018
32.6.1 Binäre Suche	1021
32.6.2 merge()	1022
32.6.3 Mengenalgorithmen	1023
32.6.4 Heaps	1024
32.6.5 lexicographical_compare()	1026
32.7 Minimum und Maximum	1026
32.8 Ratschläge	1028
33 STL-Iteratoren	1029
33.1 Einführung	1029
33.1.1 Iteratormodell	1029
33.1.2 Iteratorkategorien	1031
33.1.3 Iterator-Traits	1032
33.1.4 Iteratoroperationen	1035
33.2 Iteratoradapter	1036
33.2.1 Reverse-Iteratoren	1036
33.2.2 Einfügeiteratoren	1039
33.2.3 Verschiebeiteratoren	1040
33.3 Bereichszugriffsfunktionen	1041
33.4 Funktionsobjekte	1042
33.5 Funktionsadapter	1043

33.5.1	bind()	1044
33.5.2	mem_fn()	1046
33.5.3	function	1046
33.6	Ratschläge	1049
34	Speicher und Ressourcen	1051
34.1	Einführung	1051
34.2	„Beinahe-Container“	1051
34.2.1	array	1052
34.2.2	bitset	1055
34.2.2.1	Konstruktoren	1056
34.2.2.2	bitset -Operationen	1058
34.2.3	vector<bool>	1060
34.2.4	Tupel	1061
34.2.4.1	pair	1061
34.2.4.2	tuple	1063
34.3	Ressourcenverwaltungszeiger	1065
34.3.1	unique_ptr	1066
34.3.2	shared_ptr	1069
34.3.3	weak_ptr	1073
34.4	Allokatoren	1076
34.4.1	Der Standard-Allokator	1077
34.4.2	Allokator-Traits	1079
34.4.3	Zeiger-Traits	1080
34.4.4	Allokatoren mit eigenem Gültigkeitsbereich	1080
34.5	Die Schnittstelle zur Garbage Collection	1082
34.6	Nichtinitialisierter Speicher	1085
34.6.1	Temporäre Puffer	1086
34.6.2	raw_storage_iterator	1086
34.7	Ratschläge	1088
35	Utilities	1089
35.1	Einführung	1089
35.2	Zeit	1089
35.2.1	duration	1090
35.2.2	time_point	1093
35.2.3	Zeitgeber	1095
35.2.4	Zeit-Traits	1096
35.3	Rationale Arithmetik zur Übersetzungszeit	1097
35.4	Typfunktionen	1099
35.4.1	Typ-Traits	1099
35.4.2	Typgeneratoren	1104
35.5	Kleinere Utilities	1109

35.5.1	move() and forward()	1109
35.5.2	swap()	1110
35.5.3	Relationale Operatoren	1111
35.5.4	Vergleichen und Hashing von type_info	1112
35.6	Ratschläge	1113
36	Strings	1115
36.1	Einführung	1115
36.2	Zeichenklassifizierung	1115
36.2.1	Klassifizierungsfunktionen	1115
36.2.2	Zeichen-Traits	1117
36.3	Strings	1118
36.3.1	string im Vergleich zu C-Strings	1119
36.3.2	Konstruktoren	1121
36.3.3	Fundamentale Operationen	1123
36.3.4	String-Ein-/Ausgabe	1125
36.3.5	Numerische Konvertierungen	1125
36.3.6	STL-ähnliche Operationen	1127
36.3.7	Die find -Familie	1130
36.3.8	Teilstrings	1132
36.4	Ratschläge	1133
37	Reguläre Ausdrücke	1135
37.1	Reguläre Ausdrücke	1135
37.1.1	Notation regulärer Ausdrücke	1136
37.2	regex	1141
37.2.1	Übereinstimmungsergebnisse	1143
37.2.2	Formatierung	1146
37.3	Funktionen für reguläre Ausdrücke	1147
37.3.1	regex_match()	1147
37.3.2	regex_search()	1149
37.3.3	regex_replace()	1150
37.4	Iteratoren für reguläre Ausdrücke	1152
37.4.1	regex_iterator	1152
37.4.2	regex_token_iterator	1153
37.5	regex_traits	1156
37.6	Ratschläge	1157
38	E/A-Streams	1159
38.1	Einführung	1159
38.2	Die E/A-Stream-Hierarchie	1161
38.2.1	Datei-Streams	1162
38.2.2	String-Streams	1164

38.3	Fehlerbehandlung	1166
38.4	Ein-/Ausgabeoperationen	1168
38.4.1	Eingabeoperationen	1168
38.4.1.1	Formatierte Eingabe	1169
38.4.1.2	Unformatierte Eingabe	1170
38.4.2	Ausgabeoperationen	1171
38.4.2.1	Virtuelle Ausgabefunktionen	1173
38.4.3	Manipulatoren	1174
38.4.4	Stream-Status	1175
38.4.5	Formatierung	1179
38.4.5.1	Formatierungsstatus	1180
38.4.5.2	Standardmanipulatoren	1182
38.4.5.3	Benutzerdefinierte Manipulatoren	1185
38.5	Stream-Iteratoren	1187
38.6	Puffer	1188
38.6.1	Ausgabe-Streams und -puffer	1192
38.6.2	Eingabe-Streams und -puffer	1193
38.6.3	Pufferiteratoren	1194
38.6.3.1	istreambuf_iterator	1195
38.6.3.2	ostreambuf_iterator	1196
38.7	Ratschläge	1196
39	Locales	1199
39.1	Kulturelle Unterschiede behandeln	1199
39.2	Die Klasse locale	1202
39.2.1	Benannte Locales	1204
39.2.1.1	Neue Locales konstruieren	1207
39.2.2	Strings vergleichen	1208
39.3	Die Klasse facet	1209
39.3.1	Auf Facetten in einem Locale zugreifen	1210
39.3.2	Eine einfache benutzerdefinierte Facette	1211
39.3.3	Locales und Facetten verwenden	1214
39.4	Standardfacetten	1215
39.4.1	String-Vergleich	1217
39.4.1.1	Benannte collate -Facetten	1220
39.4.2	Numerische Formatierung	1220
39.4.2.1	Numerische Interpunktion	1221
39.4.2.2	Numerische Ausgabe	1222
39.4.2.3	Numerische Eingabe	1225
39.4.3	Formatierung von Geldbeträgen	1226
39.4.3.1	Interpunktion bei Geldbeträgen	1227
39.4.3.2	Ausgabe von Geldbeträgen	1230
39.4.3.3	Eingabe von Geldbeträgen	1231

39.4.4	Datum und Uhrzeit formatieren	1232
39.4.4.1	time_put	1232
39.4.4.2	time_get	1233
39.4.5	Zeichenklassifizierung	1235
39.4.6	Zeichencodes konvertieren	1239
39.4.7	Meldungen	1243
39.4.7.1	Meldungen von anderen Facetten verwenden	1246
39.5	Komfortschnittstellen	1247
39.5.1	Zeichenklassifizierung	1247
39.5.2	Zeichenkonvertierungen	1248
39.5.3	String-Konvertierungen	1248
39.5.4	Pufferkonvertierungen	1250
39.6	Ratschläge	1250
40	Numerische Berechnungen	1253
40.1	Einführung	1253
40.2	Numerische Grenzen	1253
40.2.1	Makros für Grenzwerte	1256
40.3	Mathematische Standardfunktionen	1257
40.4	Komplexe Zahlen	1259
40.5	Ein numerisches Array: valarray	1260
40.5.1	Konstruktoren und Zuweisungen	1261
40.5.2	Indizierung	1263
40.5.3	Operationen	1264
40.5.4	Slices	1267
40.5.5	slice_array	1269
40.5.6	Verallgemeinerte Slices	1270
40.6	Verallgemeinerte numerische Algorithmen	1271
40.6.1	accumulate()	1272
40.6.2	inner_product()	1273
40.6.3	partial_sum() und adjacent_difference()	1274
40.6.4	iota()	1275
40.7	Zufallszahlen	1275
40.7.1	Zufallszahlenmodule	1278
40.7.2	Zufallsgerät	1280
40.7.3	Verteilungen	1281
40.7.4	Zufallszahlen im Stil von C	1285
40.8	Ratschläge	1286
41	Nebenläufigkeit	1287
41.1	Einführung	1287
41.2	Speichermodelle	1289
41.2.1	Speicherstellen	1290

41.2.2	Umordnung von Befehlen	1291
41.2.3	Speicherordnung	1292
41.2.4	Data Races	1293
41.3	Atomare Datentypen	1295
41.3.1	Atomare Typen	1298
41.3.2	Flags und Fences	1303
41.3.3	Atomare Flags	1303
41.3.3.1	Fences	1304
41.4	volatile	1304
41.5	Ratschläge	1305
42	Threads und Tasks	1307
42.1	Einführung	1307
42.2	Threads	1307
42.2.1	Identität	1309
42.2.2	Konstruktion	1310
42.2.3	Zerstörung	1311
42.2.4	join()	1312
42.2.5	detach()	1313
42.2.6	Namespace this_thread	1315
42.2.7	Einen Thread vorzeitig beenden	1316
42.2.8	thread_local -Daten	1316
42.3	Data Races vermeiden	1318
42.3.1	Mutexe	1319
42.3.1.1	mutex und recursive_mutex	1320
42.3.1.2	mutex -Fehler	1322
42.3.1.3	timed_mutex und recursive_timed_mutex	1323
42.3.1.4	lock_guard und unique_lock	1324
42.3.2	Mehrere Sperren	1328
42.3.3	call_once()	1329
42.3.4	Bedingungsvariablen	1330
42.3.4.1	condition_variable_any	1335
42.4	Task-basierte Nebenläufigkeit	1336
42.4.1	future und promise	1337
42.4.2	promise	1338
42.4.3	packaged_task	1339
42.4.4	future	1342
42.4.5	shared_future	1345
42.4.6	async()	1346
42.4.7	Ein paralleles find() -Beispiel	1349
42.5	Ratschläge	1353

43 Die C-Standardbibliothek	1355
43.1 Einführung	1355
43.2 Dateien	1355
43.3 Die printf() -Familie	1356
43.4 C-Strings	1361
43.5 Speicher	1362
43.6 Datum und Uhrzeit	1364
43.7 Diverses	1367
43.8 Ratschläge	1369
44 Kompatibilität	1371
44.1 Einführung	1371
44.2 C++11-Erweiterungen	1372
44.2.1 Sprachfeatures	1372
44.2.2 Komponenten der Standardbibliothek	1373
44.2.3 Veraltete Features	1374
44.2.4 Umgang mit älteren C++-Implementierungen	1375
44.3 C/C++-Kompatibilität	1376
44.3.1 C und C++ sind Geschwister	1376
44.3.2 „Stillschweigende“ Unterschiede	1378
44.3.3 C-Code, der kein C++ ist	1379
44.3.3.1 Probleme mit „klassischem C“	1382
44.3.3.2 C-Features, die von C++ nicht übernommen wurden	1382
44.3.4 C++-Code, der kein C ist	1382
44.4 Ratschläge	1384
Index	1387