

Auf einen Blick

1	Einstieg in die Welt von C++	15
2	Verwendung der Basisdatentypen in C++	22
3	Kontrollstrukturen, Funktionen und Präprozessor-Direktiven	74
4	Höhere und fortgeschrittene Datentypen	127
5	Modularisierung	186
6	Klassen	210
7	Objekte und Klassenelemente	249
8	Operatoren überladen	290
9	Vererbung (Abgeleitete Klassen)	316
10	Templates	344
11	Exception-Handling (Fehlerbehandlung)	370
12	Weitere Neuigkeiten in C++11	388

Inhalt

1	Einstieg in die Welt von C++	15
1.1	Die (Kurz-)Geschichte von C++	15
1.2	Der ANSI-C++-Standard	17
1.3	Was benötige ich für C++?	18
1.4	Welches Betriebssystem ...?	20
1.5	Was kann ich von dem Buch erwarten?	20
1.6	Listings zum Buch	20
1.7	Aufgaben im Buch	20
1.7.1	Level 1	21
1.7.2	Level 2	21
1.7.3	Level 3	21
2	Verwendung der Basisdatentypen in C++	22
2.1	Das erste Programm in C++	22
2.2	Die Standard-Eingabe- und -Ausgabestreams	24
2.2.1	Die Streams von C++	24
2.2.2	Ausgabe mit »cout«	25
2.2.3	Ausgabe mit »cerr« oder »clog«	27
2.2.4	Eingabe mit »cin«	27
2.3	Einige Symbole von C++	29
2.3.1	Bezeichner	29
2.3.2	Literale	29
2.3.3	Kommentare	31
2.4	Grundlagen zu den Basisdatentypen	32
2.4.1	Deklaration und Definition von Variablen	32
2.4.2	Initialisieren einer Variablen	34
2.4.3	Vereinheitlichte Initialisierung mit C++11	35
2.4.4	Ganzzahltypen	36
2.4.5	Vorzeichenlos und vorzeichenbehaftet	47
2.4.6	Fließkommazahlentypen	48
2.4.7	Limits für die Basisdatentypen	50

2.4.8	Die Byte-Größe mit dem »sizeof«-Operator	52
2.4.9	Sicherheit beim Kompilieren mit »static_assert« (C++11)	52
2.5	»auto«-Typ (C++11)	53
2.6	Rechnen mit Zahlen	54
2.6.1	Arithmetische Operatoren	55
2.6.2	Ungenauere Fließkommazahlen	57
2.6.3	Erweiterte Darstellung arithmetischer Operatoren	59
2.6.4	Inkrement- und Dekrementoperator	59
2.7	Zufallszahlen (C++11)	61
2.8	Typumwandlung	62
2.8.1	Implizite Umwandlung durch den Compiler	63
2.8.2	Automatische Typumwandlung beschränken (C++11)	66
2.8.3	Explizite Typumwandlung	67
2.8.4	C++-Typumwandlungs-Operatoren	67
2.8.5	Gefährliche Typumwandlung mit den alten C-Casts	70
2.9	Konstanten	71
2.10	Aufgaben	72
2.10.1	Level 1	72
2.10.2	Level 2	72
2.10.3	Level 3	73

3 Kontrollstrukturen, Funktionen und Präprozessor-Direktiven 74

3.1	Kontrollstrukturen	74
3.1.1	Anweisungen und Anweisungsblöcke	74
3.1.2	Verzweigungen	75
3.1.3	Der Bedingungsoperator ?:	83
3.1.4	Logische Operatoren	84

3.1.5	Die Fallunterscheidung – »switch«	87
3.1.6	Schleifen	90
3.2	Funktionen	100
3.2.1	Funktionen definieren	101
3.2.2	Funktionen aufrufen	102
3.2.3	Funktionen deklarieren	103
3.2.4	Funktionsparameter (Call-by-Value)	104
3.2.5	Standardparameter	105
3.2.6	Rückgabewert von Funktionen	107
3.2.7	Funktionen überladen	109
3.2.8	Gültigkeitsbereich von lokalen und globalen Variablen	111
3.2.9	Inline-Funktionen	113
3.2.10	Die »main«-Funktion	115
3.2.11	Programmende	116
3.3	Präprozessor-Direktiven	117
3.3.1	Die »#include«-Direktive	118
3.3.2	Die »#define«-Direktive	119
3.3.3	Bedingte Kompilierung	121
3.3.4	Weitere Präprozessor-Direktiven	123
3.4	Aufgaben	124
3.4.1	Level 1	124
3.4.2	Level 2	124
3.4.3	Level 3	126
4	Höhere und fortgeschrittene Datentypen	127
4.1	Zeiger	127
4.1.1	Zeiger deklarieren	128
4.1.2	Adresse im Zeiger speichern	129
4.1.3	Zeiger dereferenzieren	131
4.1.4	Speichergröße von Zeigern	134
4.1.5	Zeigerarithmetik	134
4.2	Referenzen	135
4.3	Arrays	137
4.3.1	Der C++-Container »vector«	138
4.3.2	C-Arrays	142

4.4	Zeichenketten verwenden	146
4.4.1	Der C++-Container »string«	146
4.4.2	Unterstützung von Unicode (C++11)	148
4.4.3	C-Zeichenketten	150
4.5	Höhere Datentypen bei Funktionen verwenden	153
4.5.1	Zeiger als Funktionsparameter	153
4.5.2	Zeiger als Rückgabewert	155
4.5.3	Referenzen als Funktionsparameter	157
4.5.4	Referenzen als Rückgabewert	158
4.5.5	C-Arrays oder C-Strings als Funktionsparameter	160
4.6	Dynamische Speicherobjekte	163
4.6.1	Dynamisch Objekte mit »new« anlegen	164
4.6.2	Fehler bei der Speichieranforderung abfangen	165
4.6.3	Speicher mit »delete« wieder freigeben	166
4.7	Strukturen	168
4.7.1	Strukturen deklarieren	169
4.7.2	Zugriff auf die Strukturelemente	170
4.7.3	Zugriff auf die Elemente über Strukturzeiger	171
4.7.4	Strukturen vergleichen	173
4.7.5	Dynamische Datenstrukturen mit Strukturen	173
4.8	Union	177
4.9	Aufzählungstyp »enum«	178
4.10	Eigene Namen mit »typedef«	181
4.11	Aufgaben	182
4.11.1	Level 1	183
4.11.2	Level 2	183
4.11.3	Level 3	185
5	Modularisierung	186
5.1	Namensräume	186
5.1.1	Neuen Namensbereich deklarieren	186
5.1.2	Namensbereich verschachteln	189

5.1.3	Namensbereich ist ein eigener Gültigkeitsbereich	189
5.1.4	Namensbereich mit »using« importieren	192
5.1.5	Einzelne Bezeichner mit »using« importieren	192
5.1.6	Aliasse für Namensbereiche	193
5.1.7	Namensraum »std«	193
5.2	Speicherklassenattribute	195
5.2.1	Schlüsselwort »extern«	195
5.2.2	Schlüsselwort »static«	196
5.3	Typqualifikatoren	199
5.3.1	Schlüsselwort »const«	199
5.3.2	Schlüsselwort »volatile«	200
5.4	Weitere Attribute	201
5.5	Modulare Programmierung	201
5.5.1	Aufteilung	203
5.5.2	Die öffentliche Schnittstelle (Headerdatei)	204
5.5.3	Private Datei(en)	206
5.5.4	Die Client-Datei	206
5.5.5	Nur Objektcode oder Bibliothek vorhanden	207
5.6	Aufgaben	207
5.6.1	Level 1	208
5.6.2	Level 2	208
6	Klassen	210
6.1	Abstraktionsmechanismus	210
6.2	Klassen	211
6.2.1	Klassendefinition	211
6.2.2	Elementfunktionen definieren	213
6.2.3	Zugriffskontrolle mit »public« und »private«	215
6.2.4	Zugriff auf die Daten innerhalb einer Klasse	217

6.2.5	Objekte erzeugen und benutzen	218
6.3	Konstruktoren	223
6.3.1	Konstruktoren deklarieren	224
6.3.2	Konstruktoren definieren	225
6.3.3	Implizite Konvertierungen verhindern – »explicit«	227
6.3.4	Konstruktoren delegieren (C++11)	229
6.3.5	Standardkonstruktor (Default- Konstruktor)	231
6.3.6	Kopierkonstruktor	232
6.3.7	Move-Konstruktor (C++11)	234
6.4	Destruktoren	236
6.4.1	Destruktor deklarieren	236
6.4.2	Destruktor definieren	237
6.5	Elementfunktionen	240
6.5.1	Inline-Elementfunktionen	240
6.5.2	Konstante Elementfunktionen (read-only)	243
6.5.3	»this«-Zeiger	245
6.6	Aufgaben	246
6.6.1	Level 1	247
6.6.2	Level 2	247
6.6.3	Level 3	248
7	Objekte und Klassenelemente	249
7.1	Konstante Objekte	249
7.2	Objekte als (Element-)Funktionsargumente	250
7.2.1	Hilfsfunktionen	250
7.2.2	Wertübergabe als Kopie (Call-by-Value)	251
7.2.3	Wertübergabe als Zeiger	253
7.2.4	Wertübergabe als Referenz	255
7.2.5	Wertübergabe bei Elementfunktionen	256
7.2.6	»this«-Zeiger	257
7.3	Objekte als Rückgabewerte	260

7.4	Arrays von Objekten	262
7.5	Dynamische Objekte	263
7.6	Mehr zu den Klassenelementen	265
7.6.1	Dynamische Klassenelemente	265
7.6.2	Statische Klassenelemente	273
7.6.3	Konstante Klassenelemente	276
7.7	Andere Klassenobjekte als Datenelement einer Klasse	277
7.8	Elementinitialisierer	283
7.9	Gute Freunde (»friend«)	285
7.10	Aufgaben	287
7.10.1	Level 1	288
7.10.2	Level 2	288
7.10.3	Level 3	289
8	Operatoren überladen	290
8.1	Schlüsselwort »operator«	291
8.2	Zweistellige (arithmetische) Operatoren überladen	293
8.2.1	Operatorüberladung als Elementfunktion einer Klasse	294
8.2.2	Operatorüberladung als globale Hilfsfunktion	297
8.2.3	Zweistellige Operatoren zusammengefasst	299
8.3	Einstellige Operatoren überladen	299
8.4	Zuweisungsoperator überladen	303
8.5	Ein-/Ausgabeoperator überladen	305
8.5.1	Eingabeoperator >> überladen	306
8.5.2	Ausgabeoperator << überladen	307
8.6	Weitere Operatorüberladungen	308
8.7	Konvertierungsoperatoren	309
8.7.1	Konvertierungskonstruktor	310
8.7.2	Globale Konvertierungsfunktion	311

8.8	Aufgaben	314
8.8.1	Level 1	314
8.8.2	Level 2	314
8.8.3	Level 3	315
9	Vererbung (Abgeleitete Klassen)	316
9.1	Die Vorbereitung	317
9.2	Die Ableitung einer Klasse	318
9.2.1	»public«-Zugriffsrechte einer abgeleiteten Klasse	320
9.2.2	Erben und erweitern	320
9.2.3	Zugriff auf die Daten	321
9.2.4	Redefinition von Elementfunktionen	323
9.2.5	Konstruktoren	324
9.2.6	Destruktor	326
9.2.7	Programmbeispiel	326
9.2.8	Zugriffsrecht »protected«	327
9.2.9	Zugriffsrechte bei der Vererbung von Klassen	328
9.2.10	Implizite Typumwandlung abgeleiteter Klassen	330
9.2.11	Konstruktoren vererben (C++11)	331
9.3	Mehrfachvererbung	332
9.4	Virtuelle Vererbung	336
9.5	Aufgaben	341
9.5.1	Level 1	341
9.5.2	Level 2	342
9.5.3	Level 3	343
10	Templates	344
10.1	Funktions-Templates	344
10.1.1	Funktions-Template definieren	345
10.1.2	Typübereinstimmung	347
10.1.3	Funktions-Templates über mehrere Module	348
10.1.4	Funktions-Template spezialisieren	348

10.1.5	Templates mit verschiedenen Parametern	349
10.1.6	Explizite Template-Argumente	351
10.2	Klassen-Templates	352
10.2.1	Klassen-Template definieren	353
10.2.2	Elementfunktionen von Klassen-Templates definieren	354
10.2.3	Klassen-Template instantiieren	357
10.3	Templates der Standardbibliothek	360
10.3.1	Container der Standardbibliothek	361
10.3.2	Iteratoren	365
10.3.3	Algorithmen	365
10.4	Aufgaben	367
10.4.1	Level 1	367
10.4.2	Level 2	367
10.4.3	Level 3	368

11 Exception-Handling (Fehlerbehandlung) 370

11.1	Exception auslösen	371
11.2	Exception auffangen und behandeln	371
11.2.1	»terminate«-Handler einrichten	375
11.2.2	Alternatives »catch (...)«	376
11.2.3	Stack-Abwicklung	376
11.3	Ausnahme-Klassen (Fehlerklassen)	377
11.4	Standard-Exceptions	379
11.4.1	Virtuelle Methode »what()«	380
11.4.2	Anwenden der Standard-Exceptions	380
11.5	System-Exceptions	384
11.5.1	»bad_alloc«	384
11.5.2	»bad_cast«	384
11.5.3	»bad_typeid«	384
11.5.4	»bad_exception«	385
11.6	»noexcept« (C++11)	385
11.7	Aufgaben	387
11.7.1	Level 1	387

12 Weitere Neuigkeiten in C++11	388
12.1 Grundsätzliche Neuerungen in der Kernsprache	388
12.1.1 Range-based »for«-Schleife	388
12.1.2 Lambda-Funktionen	389
12.1.3 RValue (neue Move-Semantik)	390
12.1.4 Generische Programmierung – Variadic Templates	391
12.1.5 »decltype« und die neue Rückgabesyntax	393
12.1.6 »constexpr«	396
12.2 Standardbibliothek – neue Bibliotheken	397
12.2.1 Reguläre Ausdrücke	397
12.2.2 Zeitbibliothek	399
12.2.3 Smart Pointer	403
12.3 Multithreading	406
12.3.1 Einfache Threads erzeugen	407
12.3.2 Chaos vermeiden	409
Lösungen der Übungsaufgaben	412
Lösungen zu Kapitel 2	412
Lösungen zu Kapitel 3	414
Lösungen zu Kapitel 4	417
Lösungen zu Kapitel 5	421
Lösungen zu Kapitel 6	423
Lösungen zu Kapitel 7	427
Lösungen zu Kapitel 8	431
Lösungen zu Kapitel 9	434
Lösungen zu Kapitel 10	437
Lösungen zu Kapitel 11	439
Index	441