

Inhalt

1	Einstieg in die Welt von C++	18
<hr/>		
1.1	Der C++-Standard	18
1.2	Die nötigen Werkzeuge für C++	19
1.3	Übersetzen mit einer Entwicklungsumgebung	22
1.4	Übersetzen mit g++ und clang++	26
1.5	Listings zum Buch	28
1.6	Kontrollfragen und Aufgaben im Buch	28
1.7	Aufgabe	28
2	Erste Schritte in C++	30
<hr/>		
2.1	Das erste Programm in C++	30
2.2	Anweisungen und Ausdrücke	32
2.2.1	Anweisungen	32
2.2.2	Ausdrücke	33
2.3	Die Standard-Eingabe- und -Ausgabestreams	34
2.3.1	Die Streams von C++	34
2.3.2	Ausgabe mit »std::cout«	35
2.3.3	Ausgabe mit »std::cerr« oder »std::clog«	35
2.3.4	Eingabe mit »std::cin«	36
2.4	Einige Symbole von C++	37
2.4.1	Bezeichner	38
2.4.2	Literale	38
2.4.3	Kommentare	42
2.5	Kontrollfragen und Aufgaben	43

3	Die eingebauten C++-Basisdatentypen	44
<hr/>		
3.1	Variablen	44
3.2	Deklaration und Definition von Variablen	45
3.3	Initialisierung und Zuweisung von Werten	46
3.4	Ganzzahltypen	47
3.5	Datentyp für Zeichen	51
3.6	Internationale Zeichen	55
3.6.1	Unicode mit C++	55
3.6.2	Die Unicode-Typen »char16_t« und »char32_t«	56
3.6.3	Breite Zeichen – »wchar_t«	58
3.7	Fließkommazahlentypen	59
3.8	Limits für die Basisdatentypen	61
3.9	Die Byte-Größe mit dem »sizeof«-Operator	63
3.10	Sicherheit beim Kompilieren mit »static_assert«	63
3.11	»auto«-Typ	65
3.12	Konstanten	66
3.13	Kontrollfragen und Aufgaben	67
4	Arbeiten mit den eingebauten Typen	68
<hr/>		
4.1	Arithmetische Operatoren	68
4.2	Erweiterte Darstellung arithmetischer Operatoren	72
4.3	Inkrement- und Dekrementoperator	72
4.4	Ungenaue Fließkommazahlen	75
4.5	Typumwandlung	77

4.5.1	Implizite Umwandlung durch den Compiler	77
4.5.2	Automatische Typumwandlung beschränken	80
4.5.3	Explizite Typumwandlung	81
4.6	Kontrollfragen und Aufgaben	84
5	Kontrollstrukturen	85
<hr/>		
5.1	Anweisungsblock für Kontrollstrukturen	85
5.2	Der eingebaute Datentyp »bool«	87
5.3	Bedingte Anweisung mit »if«	88
5.4	Vergleichsoperatoren	92
5.5	Alternative »else«-Verzweigung	93
5.6	Mehrfache Verzweigung	95
5.7	Der Bedingungsoperator »?:«	98
5.8	Logische Operatoren	100
5.9	Die Fallunterscheidung – »switch«	103
5.10	Die kopfgesteuerte »while«-Schleife	106
5.11	Die fußgesteuerte »do while«-Schleife	109
5.12	Die Zählschleife »for«	112
5.13	Kontrollierte Sprunganweisungen	116
5.13.1	Die »break«-Anweisung	116
5.13.2	Die »continue«-Anweisung	117
5.14	Kontrollfragen und Aufgaben	119
6	Arrays und Strings	120
<hr/>		
6.1	Arrays	120

Inhalt

6.1.1	Der C++-Container »std::vector«	121
6.1.2	C-Arrays	126
6.2	Strings verwenden	131
6.2.1	Der C++-Container »std::string«	131
6.2.2	Unterstützung von Unicode	133
6.2.3	C-Zeichenketten	134
6.3	Kontrollfragen und Aufgaben	136
7	Referenzen und Zeiger	138
<hr/>		
7.1	Referenzen	138
7.2	Zeiger	141
7.2.1	Syntax von Zeigern	142
7.2.2	Adresse im Zeiger speichern	143
7.2.3	Zeiger dereferenzieren	145
7.2.4	Zeiger prüfen	147
7.3	Kontrollfragen und Aufgaben	148
8	Funktionen	150
<hr/>		
8.1	Grundlage zu den Funktionen	150
8.1.1	Funktionen definieren	150
8.1.2	Funktionen aufrufen	151
8.1.3	Funktionen deklarieren	152
8.1.4	Funktionsparameter (Call-by-Value)	154
8.1.5	Konstante Funktionsparameter	156
8.1.6	Standardparameter	156
8.1.7	Rückgabewert von Funktionen	159
8.1.8	Funktionen überladen	161

8.1.9	Gültigkeitsbereich von lokalen und globalen Variablen	163
8.1.10	Die »main()«-Funktion	165
8.1.11	Programmende	167
8.2	Referenzen als Parameter und Rückgabe	168
8.2.1	Referenzen als Parameter	168
8.2.2	Referenzen als Rückgabe	172
8.3	Zeiger als Parameter und Rückgabewert	173
8.3.1	Referenzen vs. Zeiger als Parameter	174
8.4	Größere Daten als Funktionsparameter	175
8.5	C-Arrays oder C-Strings als Funktionsparameter	177
8.6	Kontrollfragen und Aufgaben	178
9	Präprozessor-Direktiven und Modularisierung	181
<hr/>		
9.1	Präprozessor-Direktiven	181
9.1.1	Die »#include«-Direktive	182
9.1.2	Die »#define«-Direktive	184
9.1.3	Bedingte Kompilierung	185
9.2	Namensräume	187
9.2.1	Neuen Namensbereich deklarieren	187
9.2.2	Namensbereich verschachteln	190
9.2.3	Namensbereich ist ein eigener Gültigkeitsbereich	190
9.2.4	Namensbereich mit »using« importieren	193
9.2.5	Einzelne Bezeichner mit »using« importieren	195
9.2.6	Aliasse für Namensbereiche	195
9.2.7	Anonymer Namensraum	196
9.2.8	Namensraum »std«	196
9.3	Speicherklassenattribute	198

Inhalt

9.3.1	Schlüsselwort »extern«	198
9.3.2	Schlüsselwort »static«	199
9.4	Typqualifikatoren	201
9.5	Modularisierung	202
9.5.1	Aufteilung	204
9.5.2	Die öffentliche Schnittstelle (Headerdatei)	205
9.5.3	Private Datei(en)	206
9.5.4	Die Client-Datei	207
9.5.5	Aufgabe	208
9.5.6	Nur Objektcode oder Bibliothek vorhanden	209
9.6	Kontrollfragen und Aufgaben	209
10	Dynamische Speicherobjekte und Strukturen	212
<hr/>		
10.1	Dynamische Speicherobjekte	212
10.1.1	Dynamisch Objekte mit »new« anlegen	213
10.1.2	Fehler bei der Speichieranforderung abfangen	215
10.1.3	Speicher mit »delete« wieder freigeben	215
10.1.4	Der smarte »unique_ptr«-Pointer	217
10.2	Erste eigene Datentypen mit Strukturen	219
10.2.1	Strukturen deklarieren, erzeugen und initialisieren	221
10.2.2	Zugriff auf die Strukturelemente	222
10.2.3	Zugriff auf die Elemente in einer Funktion	223
10.2.4	Methoden statt Funktionen	225
10.2.5	Strukturen vergleichen	226
10.3	Aufzählungstyp »enum«	227
10.4	Eigene Namen mit »using«	230
10.5	Kontrollfragen und Aufgaben	231

11 Klassen	232
11.1 Klassen	232
11.1.1 Klassendefinition	233
11.1.2 Methoden definieren	235
11.1.3 Zugriffskontrolle mit »public« und »private«	237
11.1.4 Zugriff auf die Daten innerhalb einer Klasse	240
11.1.5 Objekte erzeugen und benutzen	241
11.2 Konstruktoren	246
11.2.1 Konstruktoren deklarieren	247
11.2.2 Konstruktoren definieren	248
11.2.3 Implizite Konvertierungen verhindern – »explicit«	251
11.2.4 Optimierung 1: Klassenelemente gleich direkt initialisieren	252
11.2.5 Optimierung 2: Konstruktoren delegieren	253
11.2.6 Standardkonstruktor (Default-Konstruktor)	255
11.2.7 Kopierkonstruktor	256
11.2.8 Verschiebekonstruktor (Move-Konstruktor)	258
11.3 Destruktoren	260
11.3.1 Lebensdauer eines Objekts	261
11.3.2 Wann wird ein Destruktor erforderlich?	261
11.3.3 Destruktor deklarieren	262
11.3.4 Destruktor definieren	263
11.4 Methoden	266
11.4.1 »inline«-Methoden	266
11.4.2 Konstante Methoden (»nur-lesen-erlaubt«)	269
11.4.3 »this«-Zeiger	271
11.5 Kontrollfragen und Aufgaben	273

12	Objekte und Klassenelemente	275
12.1	Konstante Objekte	275
12.2	Objekt einer Klasse als Parameter	276
12.2.1	Objekte einer Klasseninstanz an eine Funktion übergeben	276
12.2.2	Objekte von Klasseninstanz an eine Methode übergeben	280
12.2.3	Das Zielobjekt mit dem »this«-Zeiger	282
12.3	Objekte einer Klasse als Rückgabewerte	284
12.4	Arrays von Objekten	287
12.5	Dynamische Objekte	288
12.6	Mehr zu den Klassenelementen	290
12.6.1	Rohe Zeiger als Klassenelemente oder direkt die Nullregel	290
12.6.2	Die großen Fünf	299
12.6.3	Statische Klassenelemente	302
12.6.4	Konstante Klassenelemente	306
12.7	Andere Klassenobjekte als Datenelement einer Klasse	308
12.8	Freundfunktionen (»friend«)	313
12.9	Kontrollfragen und Aufgaben	315
13	Operatoren überladen	317
13.1	Schlüsselwort »operator«	318
13.2	Zweistellige (arithmetische) Operatoren überladen	320
13.2.1	Operatorüberladung als Methode einer Klasse	321
13.2.2	Operatorüberladung als globale Hilfsfunktion	325

13.3 Einstellige Operatoren überladen	327
13.4 Zuweisungsoperator überladen	331
13.5 Ein-/Ausgabeoperator überladen	336
13.5.1 Eingabeoperator »>>« überladen	336
13.5.2 Ausgabeoperator »<<« überladen	337
13.6 Weitere Operatorüberladungen	339
13.7 Konvertierungsoperatoren	340
13.7.1 Konvertierungskonstruktor	340
13.7.2 Konvertierungsfunktion	342
13.8 Kontrollfragen und Aufgaben	344
 14 Vererbung (Abgeleitete Klassen)	 345
14.1 Die Vorbereitung	346
14.2 Die Ableitung einer Klasse	347
14.2.1 »public«-Zugriffsrechte einer abgeleiteten Klasse	350
14.2.2 Erben und erweitern	350
14.2.3 Zugriff auf die Daten	351
14.2.4 Überschreiben von Klassenelementen	353
14.2.5 Konstruktoren	354
14.2.6 Destruktor	356
14.2.7 Programmbeispiel	356
14.2.8 Zugriffsrecht »protected«	357
14.2.9 Implizite Typumwandlung abgeleiteter Klassen	359
14.2.10 Konstruktoren vererben	360
14.2.11 Virtuelle Methoden	361
14.2.12 Abstrakte Klassen und Methoden	364
14.3 Kontrollfragen und Aufgaben	364

15	Templates	366
<hr/>		
15.1	Funktions-Templates	366
15.1.1	Funktions-Template definieren	367
15.1.2	Typübereinstimmung	370
15.1.3	Funktions-Templates über mehrere Module	370
15.1.4	Funktions-Template spezialisieren	370
15.1.5	Templates mit verschiedenen Parametern	372
15.1.6	Explizite Template-Argumente	374
15.2	Klassen-Templates	375
15.2.1	Klassen-Template definieren	375
15.2.2	Methoden von Klassen-Templates definieren	376
15.2.3	Methoden von Klassen-Template spezialisieren	377
15.2.4	Klassen-Template instanziiieren	378
15.2.5	Klassen-Template mit mehreren formalen Parametern	380
15.3	Templates der Standardbibliothek	380
15.3.1	Container der Standardbibliothek	381
15.3.2	Iteratoren	386
15.3.3	Algorithmen	386
15.4	Kontrollfragen und Aufgaben	388
16	Ausnahmebehandlung (Fehlerbehandlung)	390
<hr/>		
16.1	Ausnahme auslösen	391
16.2	Ausnahme auffangen und behandeln	392
16.2.1	Alternatives »catch (...)«	396
16.2.2	Stack-Abwicklung	397
16.2.3	Ausnahme mit »throw« weiterwerfen	397
16.3	Ausnahmeklassen (Fehlerklassen)	400

16.4 Standardausnahmen	402
16.4.1 Virtuelle Methode »what()«	403
16.4.2 Anwenden der Standardausnahmen	403
16.5 Systemausnahmen	407
16.5.1 »bad_alloc«	407
16.5.2 »bad_cast«	408
16.5.3 »bad_typeid«	408
16.5.4 »bad_exception«	408
16.6 Kontrollfragen	409
 17 Ein-/Ausgabestreams für Dateien	410
<hr/>	
17.1 Der Umgang mit Dateien in C++	410
17.2 Verschiedene Streams für Dateien	411
17.3 Datei öffnen und schließen	411
17.3.1 Verschiedene Modi zum Öffnen von Dateien	412
17.3.2 Datei schließen	413
17.3.3 Formatiertes Lesen und Schreiben	414
17.3.4 Byteweise lesen und schreiben	415
17.3.5 Zeilenweise lesen und schreiben	417
17.3.6 Blockweise lesen und schreiben	418
17.3.7 Lese- oder Schreibeposition ändern	419
17.4 Kontrollfragen und Aufgaben	421
 Anhang	422
<hr/>	
A Weitere Features in C++11/C++14	422
A.1 Grundsätzliche Neuerungen in der Kernsprache	422
A.1.1 Rangebasierte »for«-Schleife	422

Inhalt

A.1.2	Lambda-Funktionen	423
A.1.3	RValue (neue Move-Semantik)	425
A.1.4	Generische Programmierung – Variadic Templates	426
A.1.5	»decltype« und die neue Rückgabesyntax	428
A.1.6	»constexpr«	431
A.1.7	Code mit »[[deprecated]]« als veraltet kennzeichnen (C++14)	432
A.1.8	Binäres Literal (C++14)	434
A.1.9	Digitales Trennzeichen für große Zahlen (C++14)	434
A.1.10	Automatische Rückgabetypermittlung (C++14)	435
A.2	Standardbibliothek – neue Bibliotheken	435
A.2.1	Reguläre Ausdrücke	436
A.2.2	Zeitbibliothek	438
A.2.3	Smart Pointer	441
A.3	Multithreading	445
A.3.1	Einfache Threads erzeugen	446
A.3.2	Chaos vermeiden	448
B	Lösungen der Übungsaufgaben	452
B.1	Lösungen zu Kapitel 2	452
B.2	Lösungen zu Kapitel 3	452
B.3	Lösungen zu Kapitel 4	453
B.4	Lösungen zu Kapitel 5	455
B.5	Lösungen zu Kapitel 6	456
B.6	Lösungen zu Kapitel 7	457
B.7	Lösungen zu Kapitel 8	457
B.8	Lösungen zu Kapitel 9	460
B.9	Lösungen zu Kapitel 10	462
B.10	Lösungen zu Kapitel 11	464
B.11	Lösungen zu Kapitel 12	465

	Inhalt
B.12 Lösungen zu Kapitel 13	467
B.13 Lösungen zu Kapitel 14	468
B.14 Lösungen zu Kapitel 15	470
B.15 Lösungen zu Kapitel 16	471
B.16 Lösungen zu Kapitel 17	471
 Index	 473