

## Auf einen Blick

<b>TEIL I</b>	
Erste Schritte .....	23
<b>TEIL II</b>	
Die Elemente von C++ .....	87
<b>TEIL III</b>	
Datenstrukturen .....	223
<b>TEIL IV</b>	
Fortgeschrittene Themen .....	365

# Inhalt

Geleitwort des Fachgutachters .....	18
Vorwort .....	20

## TEIL I Erste Schritte

### **1 Über dieses Buch** 25

---

<b>1.1 Der C++-Standard</b> .....	26
<b>1.2 Verwendete Formatierungen</b> .....	27

### **2 Vom Problem zum Programm** 29

---

<b>2.1 Was ist Programmieren?</b> .....	30
<b>2.2 Softwareentwicklungsmethoden</b> .....	30
<b>2.3 Entwurfsmuster</b> .....	32
<b>2.4 Algorithmen</b> .....	33
<b>2.5 Ressourcen</b> .....	34

### **3 Programmieren in C++** 36

---

<b>3.1 Übersetzen</b> .....	37
<b>3.2 Aktuelle Compiler</b> .....	37
3.2.1 Gnu C++ .....	39
3.2.2 Clang++ der LLVM .....	39
3.2.3 Microsoft Visual Studio .....	39
<b>3.3 Entwicklungsumgebungen</b> .....	39
<b>3.4 Die Kommandozeile unter Ubuntu</b> .....	41
3.4.1 Ein Programm erstellen .....	42
3.4.2 Automatisieren mit Makefile .....	44
<b>3.5 Die IDE »Microsoft Visual Studio Express« unter Windows</b> .....	44

3.6	Schneller .....	47
3.7	Aufgaben .....	47
<b>4</b>	<b>Ein ganz schneller Überblick</b> .....	<b>49</b>
4.1	Kommentare .....	50
4.2	Die »include«-Direktive .....	50
4.3	Die Standardbibliothek .....	51
4.4	Die Funktion »main()« .....	51
4.5	Typen .....	51
4.6	Variablen .....	52
4.7	Initialisierung .....	52
4.8	Ausgabe auf der Konsole .....	53
4.9	Anweisungen .....	53
4.10	Aufgaben .....	54
<b>5</b>	<b>Ohne Eile erklärt</b> .....	<b>56</b>
5.1	Leerräume, Bezeichner und Token .....	58
5.2	Kommentare .....	59
5.3	Funktionen und Argumente .....	60
5.4	Seiteneffekt-Operatoren .....	61
5.5	Die »main«-Funktion .....	62
5.6	Anweisungen .....	64
5.7	Ausdrücke .....	66
5.8	Zuweisungen .....	68
5.9	Typen .....	69
5.9.1	Der Typ »int« .....	71
5.9.2	Der Typ »bool« .....	72
5.9.3	Die Typen »const char*« und »std::string« .....	72
5.9.4	Parametrisierte Typen .....	73
5.10	Variablen – Deklaration, Definition und Initialisierung .....	74
5.11	Details zur »include«-Direktive .....	75

5.12	Eingabe und Ausgabe .....	76
5.13	Der Namensraum »std« .....	77
5.14	Aufgaben .....	79
<b>6</b>	<b>Programmiertechnik, 1. Dan: Lesbarer Code</b> .....	<b>81</b>
6.1	Kommentare .....	81
6.2	Dokumentation .....	82
6.3	Einrückungen und Zeilenlänge .....	83
6.4	Zeilen pro Funktion und Datei .....	84
6.5	Klammern und Leerzeichen .....	84
6.6	Namen .....	86
<b>TEIL II Die Elemente von C++</b>		
<b>7</b>	<b>Operatoren</b> .....	<b>89</b>
7.1	Operatoren und Operanden .....	90
7.2	Überblick über Operatoren .....	90
7.3	Arithmetische Operatoren .....	91
7.4	Bitweise Arithmetik .....	92
7.5	Zuweisungsoperatoren .....	93
7.6	Post- und Präinkrement sowie Post- und Prädekrement .....	94
7.7	Relationale Operatoren .....	95
7.8	Logische Operatoren .....	95
7.8.1	Kurzschluss-Auswertung .....	95
7.8.2	Alternative Token .....	96
7.9	Pointer- und Dereferenzierungsoperatoren .....	97
7.10	Besondere Operatoren .....	97
7.11	Funktionsähnliche Operatoren .....	99
7.12	Operatorreihenfolge .....	100
7.13	Aufgaben .....	101

<b>8</b>	<b>Eingebaute Typen</b>	103
<b>8.1</b>	<b>Eingebaute Datentypen</b>	105
<b>8.2</b>	<b>Eingebaute Datentypen initialisieren</b>	105
<b>8.3</b>	<b>Ein schneller Überblick</b>	106
<b>8.4</b>	<b>Ganzzahlen</b>	107
8.4.1	Operationen auf Ganzzahlen	108
8.4.2	Arithmetische Typumwandlung	109
8.4.3	Ganzzahl-Überlauf	110
8.4.4	Ganzzahl-Literale	111
8.4.5	Alias-Zahlentypen	112
8.4.6	Fließkomma-Literale	115
8.4.7	Die Fließkomma-Besonderheiten	116
8.4.8	Fließkomma-Interna	117
<b>8.5</b>	<b>Wahrheitswerte</b>	118
<b>8.6</b>	<b>Zeichentypen</b>	119
8.6.1	Internationale Zeichen	121
8.6.2	Unicode in C++	122
<b>8.7</b>	<b>Aufgaben</b>	122
<b>9</b>	<b>Strings und Streams</b>	124
<b>9.1</b>	<b>Der Zeichenkettentyp »string«</b>	124
9.1.1	Initialisierung	126
9.1.2	Funktionen und Methoden	127
9.1.3	Andere Stringtypen	127
<b>9.2</b>	<b>Streams</b>	129
<b>9.3</b>	<b>Eingabe- und Ausgabeoperatoren</b>	130
9.3.1	»getline«	131
9.3.2	Dateien für die Ein- und Ausgabe	131
9.3.3	Manipulatoren	132
9.3.4	Der Manipulator »endl«	133
<b>9.4</b>	<b>Aufgaben</b>	134

<b>10</b>	<b>Behälter und Zeiger</b>	135
<b>10.1</b>	<b>Parametrisierte Typen</b>	136
<b>10.2</b>	<b>Die einfachen Sequenzcontainer</b>	136
10.2.1	»array«	137
10.2.2	»vector«	139
<b>10.3</b>	<b>Weitere Container</b>	141
10.3.1	Sequenzbasierte Container	141
10.3.2	Assoziative Container	142
<b>10.4</b>	<b>Container-Gemeinsamkeiten</b>	144
<b>10.5</b>	<b>Algorithmen</b>	145
<b>10.6</b>	<b>Zeiger und C-Arrays</b>	146
10.6.1	Zeigertypen	146
10.6.2	C-Arrays	146
<b>10.7</b>	<b>Aufgaben</b>	147
<b>11</b>	<b>Funktionen</b>	148
<b>11.1</b>	<b>Deklaration und Definition einer Funktion</b>	149
<b>11.2</b>	<b>Funktionsstyp</b>	150
<b>11.3</b>	<b>Funktionen verwenden</b>	150
<b>11.4</b>	<b>Eine Funktion definieren</b>	151
<b>11.5</b>	<b>Mehr zu Parametern</b>	152
11.5.1	Call-by-Value	153
11.5.2	Call-by-Reference	153
11.5.3	Konstante Referenzen	154
11.5.4	Aufruf als Wert, Referenz oder konstante Referenz?	155
<b>11.6</b>	<b>Funktionskörper</b>	156
<b>11.7</b>	<b>Parameter umwandeln</b>	158
<b>11.8</b>	<b>Funktionen überladen</b>	160
<b>11.9</b>	<b>Default-Parameter</b>	162

<b>11.10</b>	<b>Beliebig viele Argumente</b>	163
<b>11.11</b>	<b>Alternative Schreibweise zur Funktionsdeklaration</b>	164
<b>11.12</b>	<b>Spezialitäten</b>	165
11.12.1	noexcept	165
11.12.2	constexpr	165
11.12.3	Gelöschte Funktionen	166
11.12.4	Spezialitäten bei Klassenmethoden	166
<b>11.13</b>	<b>Aufgaben</b>	167
 <b>12 Anweisungen im Detail</b>		169
<b>12.1</b>	<b>Anweisungsblock</b>	171
12.1.1	Freistehende Blöcke und Gültigkeit von Variablen	172
<b>12.2</b>	<b>Die leere Anweisung</b>	174
<b>12.3</b>	<b>Deklarationsanweisung</b>	175
<b>12.4</b>	<b>Ausdrucksanweisung</b>	176
<b>12.5</b>	<b>Die if-Anweisung</b>	176
<b>12.6</b>	<b>»while«-Schleife</b>	179
<b>12.7</b>	<b>»do-while«-Schleife</b>	180
<b>12.8</b>	<b>»for«-Schleife</b>	181
<b>12.9</b>	<b>Die bereichsbasierte »for«-Schleife</b>	182
<b>12.10</b>	<b>Die »switch«-Verzweigung</b>	183
<b>12.11</b>	<b>»break«-Anweisung</b>	187
<b>12.12</b>	<b>Die »continue«-Anweisung</b>	188
<b>12.13</b>	<b>Die »return«-Anweisung</b>	189
<b>12.14</b>	<b>Die »goto«-Anweisung</b>	190
<b>12.15</b>	<b>»try-catch«-Block und »throw«</b>	192
<b>12.16</b>	<b>Zusammenfassung</b>	193
<b>12.17</b>	<b>Aufgaben</b>	193

<b>13</b>	<b>Ausdrücke im Detail</b>	196
<b>13.1</b>	<b>Berechnungen und Seiteneffekte</b>	197
<b>13.2</b>	<b>Arten von Ausdrücken</b>	198
<b>13.3</b>	<b>Literale</b>	199
<b>13.4</b>	<b>Bezeichner</b>	200
<b>13.5</b>	<b>Klammern</b>	201
<b>13.6</b>	<b>Funktionsaufruf und Index-Zugriff</b>	201
<b>13.7</b>	<b>Zuweisung</b>	201
<b>13.8</b>	<b>Typumwandlung</b>	203
<b>13.9</b>	<b>Aufgaben</b>	204
 <b>14 Fehlerbehandlung</b>		205
<b>14.1</b>	<b>Fehlerbehandlung mit Rückgabewerten</b>	207
<b>14.2</b>	<b>Was ist eine Ausnahme?</b>	210
14.2.1	Ausnahmen auslösen und behandeln	211
14.2.2	Aufrufstapel abwickeln	212
<b>14.3</b>	<b>Kleinere Fehlerbehandlungen</b>	213
<b>14.4</b>	<b>Weiterwerfen – »rethrow«</b>	213
<b>14.5</b>	<b>Die Reihenfolge im »catch«</b>	214
14.5.1	Kein »finally«	215
14.5.2	Exceptions der Standardbibliothek	215
<b>14.6</b>	<b>Typen für Exceptions</b>	216
<b>14.7</b>	<b>Wenn eine Exception aus »main« herausfällt</b>	217
<b>14.8</b>	<b>Aufgaben</b>	217
 <b>15 Programmiertechnik, 2. Dan: Modularisierung</b>		219
<b>15.1</b>	<b>Programm, Bibliothek, Objektdatei</b>	219
<b>15.2</b>	<b>Bausteine</b>	220
<b>15.3</b>	<b>Trennen der Funktionalitäten</b>	221

## TEIL III Datenstrukturen

<b>16 Erste eigene Datentypen</b>	225
<b>16.1 Initialisierung</b>	226
<b>16.2 Rückgabe eigener Typen</b>	227
<b>16.3 Methoden statt Funktionen</b>	228
<b>16.4 Das bessere »drucke«</b>	231
<b>16.5 Eine Ausgabe wie jede andere</b>	232
<b>16.6 Methoden inline definieren</b>	233
<b>16.7 Implementierung und Definition trennen</b>	234
<b>16.8 Initialisierung per Konstruktor</b>	235
16.8.1 Member-Defaultwerte in der Deklaration	238
16.8.2 Konstruktor-Delegation	238
16.8.3 Default-Werte für die Konstruktor-Parameter	239
16.8.4 »init«-Methode nicht im Konstruktor aufrufen	240
16.8.5 Exceptions im Konstruktor	241
<b>16.9 Struktur oder Klasse?</b>	241
16.9.1 Kapselung	242
16.9.2 Public und Private, Struktur und Klasse	242
16.9.3 Daten mit »struct«, Verhalten mit »class«	243
16.9.4 Initialisierung von Typen mit privaten Daten	243
<b>16.10 Zusammenfassung</b>	245
<b>16.11 Aufgaben</b>	245
<b>17 Verwendung eigener Datentypen</b>	248
<b>17.1 Klassen als Werte verwenden</b>	251
17.1.1 Wert-Parameter	251
17.1.2 Rückgaben	252
17.1.3 Performance beim Zurückgeben	253
<b>17.2 Konstruktoren nutzen</b>	253
<b>17.3 Typumwandlungen</b>	254

<b>17.4 Kapseln und entkapseln</b>	256
17.4.1 Entkapseln	256
17.4.2 Setzen Sie Konvertierungsmethoden sparsam ein	257
17.4.3 Totale Kapselung	258
17.4.4 Flüssiges Programmieren	259
17.4.5 Methoden mit »const« markieren	260
<b>17.5 Typen lokal einen Namen geben</b>	260
<b>17.6 Typdeduktion mit »auto«</b>	263
<b>17.7 Eigene Klassen in Standardcontainern</b>	266
<b>17.8 Aufgaben</b>	268
<b>18 Namespace und Static</b>	270
<b>18.1 Der Namensraum »std«</b>	270
<b>18.2 Anonymer Namensraum</b>	274
<b>18.3 »static« macht lokal</b>	275
<b>18.4 »static« teilt gern</b>	276
<b>18.5 »static« macht dauerhaft</b>	279
<b>18.6 Zusammenfassung</b>	281
<b>18.7 Aufgaben</b>	281
<b>19 Const</b>	284
<b>19.1 Const-Parameter</b>	285
<b>19.2 Const-Methoden</b>	286
<b>19.3 Const-Variablen</b>	288
<b>19.4 Const-Rückgaben</b>	289
19.4.1 Const zusammen mit static	292
19.4.2 Noch konstanter mit »constexpr«	293
19.4.3 »constexpr« als Rückgabe	295
<b>19.5 Const-Korrektheit</b>	296
<b>19.6 Zusammenfassung</b>	297
<b>19.7 Aufgaben</b>	298

<b>20 Vererbung</b>	300
<b>20.1 Beziehungen</b>	301
20.1.1 Hat-ein-Komposition	301
20.1.2 Hat-ein-Aggregation	301
20.1.3 Ist-ein-Vererbung	302
20.1.4 Nicht: ist eine Instanz von	302
<b>20.2 Vererbung in C++</b>	303
<b>20.3 Hat-ein versus ist-ein</b>	304
<b>20.4 Gemeinsamkeiten finden</b>	304
<b>20.5 Abgeleitete Typen erweitern</b>	307
<b>20.6 Methoden überschreiben</b>	308
<b>20.7 Wie Methoden funktionieren</b>	309
<b>20.8 Virtuelle Methoden</b>	310
<b>20.9 Konstruktoren in Klassenhierarchien</b>	312
<b>20.10 Typumwandlung in Klassenhierarchien</b>	314
20.10.1 Die Vererbungshierarchie aufwärts umwandeln	314
20.10.2 Die Vererbungshierarchie abwärts umwandeln	314
20.10.3 Referenzen behalten auch die Typinformation	315
<b>20.11 Wann virtuell?</b>	315
<b>20.12 Andere Designs zur Erweiterbarkeit</b>	317
<b>20.13 Aufgaben</b>	318
<b>21 Der Lebenszyklus von Klassen</b>	321
<b>21.1 Erzeugung und Zerstörung</b>	322
<b>21.2 Temporary: Kurzlebige Werte</b>	324
<b>21.3 Der Destruktor zum Konstruktor</b>	325
21.3.1 Kein Destruktor nötig	327
21.3.2 Ressourcen im Destruktor	328
<b>21.4 Yoda-Bedingung</b>	330
<b>21.5 Konstruktion, Destruktion und Exceptions</b>	331
<b>21.6 Kopieren</b>	333
<b>21.7 Zuweisungsoperator</b>	335

<b>21.8 Streichen von Methoden</b>	339
<b>21.9 Verschiebeoperationen</b>	340
<b>21.10 Operatoren</b>	344
<b>21.11 Eigene Operatoren in einem Datentyp</b>	348
<b>21.12 Besondere Klassenformen</b>	353
21.12.1 Abstrakte Klassen und Methoden	353
21.12.2 Aufzählungsklassen	355
<b>21.13 Aufgaben</b>	356
<b>22 Programmieretechnik, 3. Dan: Die Nuller-Regel</b>	359
<b>22.1 Die großen Fünf</b>	359
<b>22.2 Hilfskonstrukt per Verbot</b>	360
<b>22.3 Die Nullerregel und ihr Einsatz</b>	361
<b>22.4 Ausnahmen von der Nullerregel</b>	362
<b>TEIL IV Fortgeschrittene Themen</b>	
<b>23 Zeiger</b>	367
<b>23.1 Adressen</b>	368
<b>23.2 Zeiger</b>	369
<b>23.3 Heapspeicher und Stapelspeicher</b>	372
23.3.1 Der Stapel	372
23.3.2 Der Heap	374
<b>23.4 Smarte Pointer</b>	376
23.4.1 »unique_ptr«	378
23.4.2 »shared_ptr«	382
<b>23.5 Rohe Zeiger</b>	385
<b>23.6 C-Arrays</b>	390
23.6.1 Rechnen mit Zeigern	391
23.6.2 Verfall von C-Arrays	392
23.6.3 Dynamische C-Arrays	393
23.6.4 Zeichenkettenliterale	394

<b>23.7 Iteratoren</b> .....	396
23.7.1 Mehr Funktionalität mit Iteratoren .....	398
23.7.2 Zeiger als Iteratoren .....	399
<b>23.8 Zeiger im Container</b> .....	400
<b>23.9 Die Ausnahme: Wann das Wegräumen nicht nötig ist</b> .....	400
<b>23.10 Aufgaben</b> .....	402
<b>24 Makros</b> .....	405
<hr/>	
<b>24.1 Der Präprozessor</b> .....	406
<b>24.2 Vorsicht vor fehlenden Klammern</b> .....	410
<b>24.3 Vorsicht vor Mehrfachausführung</b> .....	410
<b>24.4 Typvariabilität von Makros</b> .....	411
<b>24.5 Zusammenfassung</b> .....	413
<b>24.6 Aufgaben</b> .....	414
<b>25 Schnittstelle zu C</b> .....	416
<hr/>	
<b>25.1 Mit Bibliotheken arbeiten</b> .....	417
<b>25.2 C-Header</b> .....	418
<b>25.3 C-Ressourcen</b> .....	421
<b>25.4 »void«-Pointer</b> .....	422
<b>25.5 Daten lesen</b> .....	423
<b>25.6 Das Hauptprogramm</b> .....	424
<b>25.7 Zusammenfassung</b> .....	425
<b>25.8 Aufgaben</b> .....	425

<b>26 Template-Funktionen</b> .....	427
<hr/>	
<b>26.1 Überladung</b> .....	428
<b>26.2 Ein Typ als Parameter</b> .....	429
<b>26.3 Funktionskörper einer Template-Funktion</b> .....	429
<b>26.4 Zahlen als Template-Parameter</b> .....	431
<b>26.5 Viele Funktionen</b> .....	432
<b>26.6 Parameter mit Extras</b> .....	432
<b>26.7 Template-Methoden sind auch nur Funktionen</b> .....	435
<b>26.8 Template-Funktionen in der Standardbibliothek</b> .....	436
<b>26.9 Iteratoren statt Container als Template-Parameter</b> .....	437
<b>26.10 Beispiel: Informationen über Zahlen</b> .....	439
<b>26.11 Aufgaben</b> .....	440
<b>27 Eine Klasse als Funktion</b> .....	442
<hr/>	
<b>27.1 Werte für einen »function«-Parameter</b> .....	443
<b>27.2 C-Funktionspointer</b> .....	444
<b>27.3 Die etwas andere Funktion</b> .....	445
<b>27.4 Praktische Funktoren</b> .....	448
<b>27.5 Algorithmen mit Funktoren</b> .....	450
<b>27.6 Anonyme Funktionen alias Lambda-Ausdrücke</b> .....	451
27.6.1 Lambdas mit Zugriff nach außen .....	452
27.6.2 Für Ihre Bequemlichkeit .....	454
<b>27.7 Aufgaben</b> .....	455
<b>Anhang</b> .....	457
<hr/>	
<b>A C++11-Besonderheiten</b> .....	458
<b>B Operator-Präzedenzen</b> .....	470
<b>C Lösungen</b> .....	472
<b>Index</b> .....	515