

# Inhaltsverzeichnis

Über die Autoren	7
<b>Einführung</b>	<b>25</b>
Sie benötigen keinerlei Erfahrung	25
Und auch die erste Wahl für erfahrene Leute!	26
Für alle Computer	26
Konventionen	27
Wie das Buch aufgebaut ist	27
Haufenweise Symbole	29
Wie geht es weiter?	29
Was danach kommt	30
 <b>Teil I</b>	
<b>Los geht's mit C++</b>	<b>31</b>
<b>Kapitel 1</b>	
<b>Ihr System einrichten</b>	<b>33</b>
An C++ 14 gelangen	33
Sich Code::Blocks besorgen	34
Code::Blocks installieren	35
Mit Windows arbeiten	35
Mit Mac OS/X arbeiten	37
Eine Standardinstallation unter Linux	38
Die grafische Linux-Installation verwenden	38
Die wesentlichen Programmfunktionen von Code::Blocks kennenlernen	40
Code::Blocks zum ersten Mal starten	41
Die Beispielprojekte öffnen	43
Die wichtigsten Fenster	44
Das Fenster »Start here«	44
Andere Entwicklungsumgebungen verwenden	49
 <b>Kapitel 2</b>	
<b>Die erste C++-Anwendung erstellen</b>	<b>51</b>
Ein Projekt mit Code::Blocks erstellen	51
Was sind Projekte?	51
Das erste Projekt definieren	52
Die erste Anwendung erstellen und ausführen	58

Den Code eingeben	59
Mit »main« beginnen	60
Informationen anzeigen	61
Ein bisschen Rechnen	65
Die Ausgabe mit Tabulatoren versehen	69
Lassen Sie Ihre Anwendung laufen	71

## ***Kapitel 3***

### ***Daten in C++ speichern*** **73**

Ablageorte für Daten: Variablen	73
Eine Integervariable erstellen	74
Mehrere Variablen definieren	77
Werte ändern	77
Eine Variable mit einer anderen gleichsetzen	78
Eine Variable initialisieren	79
Vergeben Sie selbst großartige Namen	80
Mit Integervariablen rechnen	81
Mit Integervariablen addieren	82
Ganzzahlige Werte subtrahieren	86
Integervariablen multiplizieren	87
Integervariablen dividieren	88
Allgemeine Zeichen oder Characters	90
Das Null-Zeichen	91
Nicht druckbare und andere Zeichen	91
Zeichenfolgen (Strings)	94
Einen Teil eines Strings auslesen	94
Einen Teil eines Strings ändern	96
Einen String erweitern	97
Zwei Strings »addieren«	97
Zwischen bedingten Operatoren entscheiden	98
Mit Boole'schen Variablen die Wahrheit sagen	100
Die Konsole auslesen	102

## ***Kapitel 4***

### ***Den Ablauf der Anwendung steuern*** **105**

Dies oder jenes tun	106
Bedingungen in C++ auswerten	106
Den richtigen C++-Operator herausfinden	107
Mehrere Bedingungen kombinieren	108
Evaluierungen in bedingte C++-Anweisungen einbinden	109
Festlegen, was wäre wenn und was ansonsten wäre	110
Mit »if« und »else« eine Etage weiter gehen	111

Aktionen mit Anweisungen wiederholen	114
Schleifenbildung	114
for-Schleifen	115
»While«-Schleifen	122
Etwas tun, während ...	125
Unterbrechen und fortfahren	126
Schleifen verschachteln	128

## **Kapitel 5**

### ***Ihr Werk mit Funktionen aufteilen*** **131**

Teilen Sie Ihr Werk auf	131
Eine Funktion aufrufen	136
Eine Variable übergeben	137
Mehrere Variablen übergeben	138
Eigene Funktionen schreiben	139
Mehrere oder keine Parameter	143
Nichts zurückgeben	145
Lokale Variablen	147
Vorwärtsreferenzen und Funktionsprototyp	148
Zwei Versionen derselben Funktion schreiben	150
Funktionen für große Strings aufrufen	152
Was ist »main()«?	154

## **Kapitel 6**

### ***Quellcodedateien aufteilen*** **159**

Mehrere Quellcodedateien erstellen	159
Ein Projekt in Code::Blocks in mehrere Quellcodedateien aufteilen	160
In Code::Blocks ein Projekt mit mehreren Dateien erstellen	163
Mehrere Quellcodedateien in anderen Compilern	165
Mehrere Dateien erstellen	166
Headerdateien gemeinsam nutzen	168
Den Header nur einmal hinzufügen	171
Spitze Klammern oder Anführungszeichen?	171
Variablen in mehreren Quellcodedateien nutzen	172
Die mysteriösen Header-Wrapper verwenden	174

## **Kapitel 7**

### ***Mit Zeigern auf die Daten verweisen*** **177**

Variablen »anhäufen« und »stapeln«	178
Die Adresse einer Variablen erhalten	181
Eine Variable mithilfe eines Zeigers ändern	183
Auf einen String zeigen	185
Auf etwas anderes zeigen	188
Tipps für Zeigervariablen	189

Dynamisches Zuweisen mit »new«	190
»new« verwenden	190
Eine Initialisierungsanweisung verwenden	192
Ein neuer String	193
Zeiger freigeben	195
Zeigervariablen an Funktionen übergeben	197
Werte von Variablen mit Zeiger ändern	197
Stringparameter ändern	201
Zeigervariablen von Funktionen zurückerhalten	202
Einen Zeiger als Nicht-Zeiger zurückgeben	205
Per Referenz übergeben	206
Als konstante Referenz übergeben	207
An die Regeln denken	208

## **Kapitel 8**

### **Mit Klassen arbeiten**

**209**

Objekte und Klassen verstehen	209
Klassen und Objekte klassifizieren	211
Mitgliedsfunktionen und Memberdaten beschreiben	212
Eine Klasse implementieren	214
Den Code von Mitgliedsfunktionen aufteilen	220
Die Bestandteile einer Klasse	223
Mit einer Klasse arbeiten	224
Auf Mitglieder zugreifen	224
Klassen und Zeiger verwenden	227
Objekte an Funktionen übergeben	230
Konstante Parameter in Funktionen verwenden	232
Den Zeiger »this« verwenden	233
Mitgliedsfunktionen überladen	236
Mit Konstruktoren und Destruktoren anfangen und aufhören	239
Mit Konstruktoren starten	239
Etwas mit Destruktoren beenden	240
Beispiele für Konstruktoren und Destruktoren	240
Den Konstruktoren Parameter hinzufügen	242
Klassenhierarchien bilden	244
Eine Hierarchie in C++ erstellen	245
Vererbungstypen verstehen	246
Aliasse von Objekten erstellen und verwenden	247

## **Kapitel 9**

### **Erweiterte C++-Funktionen nutzen**

**249**

Den Code mit Kommentaren ergänzen	249
Typen umwandeln	252
Aus der Konsole lesen	258
Die Direktiven eines Präprozessors verstehen	263
Konstanten verwenden	272
»switch«-Anweisungen verwenden	275
Aufzählungen mit Klassen überladen	278
Mit Zufallszahlen arbeiten	281
Daten in Arrays speichern	283
Ein Array aus Zeigern	284
Arrays an Funktionen übergeben	286
Zeiger addieren und subtrahieren	288

## **Teil II**

### **Objekte und Klassen verstehen**

**291**

## **Kapitel 10**

### **Objekte planen und erstellen**

**293**

Objekte erkennen	293
Die Klasse »Briefkaesten« beobachten	295
Die Klasse »Briefkasten« beobachten	297
Andere Objekte finden	298
Objekte kapseln	299
Hierarchien aufbauen	304
Eine Bibliothek einrichten	304
Mitglieder bei Vererbung schützen	305
Mitgliedsfunktionen überschreiben	311
Spezialisierung durch Polymorphie	314
Dinge abstrakt sehen	315
Klassen entdecken	317
Software entwickeln	317
Diese nervenden Klassen finden	321

## **Kapitel 11**

### **Mit Entwurfsmustern entwickeln**

**325**

Eine Einführung in einfache Muster: das Singleton	326
Eine Instanz mit einem Beobachter überwachen	329
Einen Beobachter automatisch hinzufügen	335
Mit einem Muster vermitteln	338

## **Teil III**

### **Fortgeschrittene Programmierung**

**355**

#### **Kapitel 12**

##### **Mit Arrays, Zeigern und Referenzen arbeiten**

**357**

Arrays verstehen	357
Arrays definieren	358
Arrays und Zeiger	359
Mehrdimensionale Arrays einsetzen	363
Mehrdimensionale Arrays initialisieren	364
Mehrdimensionale Arrays übergeben	365
Arrays und Befehlszeilenparameter	366
Ein Array auf dem Heap anlegen	367
Arrays aus Zeigern und Arrays aus Arrays speichern	369
Konstante Arrays erstellen	372
Mit Zeigern zeigen	372
Immer diese furchtbare Komplexität	373
Zeiger auf Funktionen	378
Eine Variable auf eine Mitgliedsfunktion zeigen lassen	379
Auf statische Mitgliedsfunktionen zeigen	382
Auf Referenzen referenzieren	382
Variablen referenzieren	383
Eine Referenz aus einer Funktion zurückgeben	384

#### **Kapitel 13**

##### **Datenstrukturen erstellen**

**387**

Mit Daten arbeiten	387
Variablen, Variablen und noch mehr Variablen	387
Variablen auf beiden Seiten der Macht	390
Ihre Daten verzaubern	391
Strukturieren Sie Ihre Daten	393
Strukturen als Komponenten-Datentypen	395
Strukturen gleichsetzen	395
Strukturtypen zurückgeben	397
Geben Sie Ihrem Raum einen Namen	398
Variablen und Teile eines Namensraums verwenden	401

## **Kapitel 14**

### **Konstruktoren, Destruktoren und Exceptions**

**403**

Objekte mit Konstruktor und Destruktor erschaffen und zerstören	403
Konstruktoren überladen	404
Mitglieder initialisieren	405
Einen Standardkonstruktor hinzufügen	410
Funktionskonstruktoren	412
Einen Konstruktor von einem anderen aus aufrufen	415
Instanzen mithilfe von Copy-Konstruktoren kopieren	416
Wenn Konstruktoren Ärger machen: Fehlerhafte Konstruktoren	419
Instanzen zerstören	419
Destruktoren virtuell erben	421
Die Ausnahme von der Regel	424
Direkte Instanzen werfen	427
Beliebige Exceptions fangen	428
Eine Exception weiterwerfen	430

## **Kapitel 15**

### **Fortgeschrittene Klassen-Techniken**

**433**

Schöner erben	433
Verwandeln Sie Ihre Vererbung	433
Die Zugriffsregeln anpassen	434
Geben Sie doch mal etwas anderes zurück	436
Mehrfachvererbung	438
Virtuelle Vererbung	441
Freundklassen und Freundfunktionen	445
Klassen und Typen in Klassen nutzen	446
Eine Klasse verschachteln	446
Typen innerhalb von Klassen	451

## **Kapitel 16**

### **Klassen mit Templates erzeugen**

**455**

Eine Klasse zum Template machen	455
Ein Template vom Code der Funktionen trennen	461
Statische Mitglieder in ein Template einsetzen	464
Ein Template parametrisieren	465
Verschiedene Parametertypen	466
Mehrere Parameter verwenden	470
Ein »typedef« für ein Template	473
Templates ableiten	474
Eine Klasse von einem Klassen-Template ableiten	475
Ein Klassen-Template von einer Klasse ableiten	477
Ein Klassen-Template von einem Klassen-Template ableiten	479

Ein Funktionstemplate erstellen	482
Funktionstemplates überladen	483
Eine Mitgliedsfunktion mit einem Template versehen	486

## **Kapitel 17**

### ***Mit der Standardbibliothek programmieren*** **489**

Aufbau der Standardbibliothek	490
Container für Ihre Klassen	490
In einem Vektor abspeichern	490
Daten per »map« ablegen	493
Instanzen, Zeiger oder Referenzen im Container verwalten	494
Instanzen vergleichen	498
Durch einen Container iterieren	504
Viele kleine Pärchen	507
Das große Container-Finale	507
Mit einem »set« zuweisen und speichern	507
Mit »set« Vereinigungs- und Schnittmengen bilden	510
Eine Liste mit »list«	513
Stapeln Sie auf der Deque	518
Mit Stacks und Queues schön in einer Reihe warten	519
Container kopieren	522
Dynamische Arrays erzeugen und einsetzen	524
Die IDE konfigurieren	524
Ein dynamisches Array deklarieren	525
Mit ungeordneten Daten arbeiten	526
Mit »std::unordered_set« ein ungeordnetes Set erstellen	527
Ungeordnete Sets bearbeiten	527

## **Kapitel 18**

### ***Mit Lambda-Ausdrücken arbeiten*** **529**

Besser lesbaren und prägnanteren C++-Code schreiben	530
Einen einfachen Lambda-Ausdruck definieren	531
Die Teile eines Lambda-Ausdrucks definieren	531
Den Compiler den Rückgabewert ermitteln lassen	532
Einen bestimmten Rückgabotyp verwenden	534
Das Schlüsselwort »auto« verwenden	535
Anwendungen mit Lambda-Ausdrücken entwickeln	538
Lambda-Ausdrücke mit mehreren Eingabeparametern erstellen	538
Mit der Capture-Klausel arbeiten	539
Daten mit einem Lambda-Ausdruck sortieren	540
Festlegen, dass der Lambda-Ausdruck Exceptions wirft	542

<b>Teil IV</b>	
<b>Probleme beheben</b>	<b>545</b>
<b>Kapitel 19</b>	
<b>Mit Bugs umgehen</b>	<b>547</b>
Das ist kein Bug, das ist ein Feature	547
Die Features einer Anwendung wie Features aussehen lassen	549
Etwas (fast) vorhersehen	550
Kurz und bündig: Fehler vermeiden	559
<b>Kapitel 20</b>	
<b>Eine Anwendung debuggen</b>	<b>561</b>
Mit Debuggern programmieren	561
Der Code::Blocks-Debugger im Überblick	564
Die Debuggingfenster des Code::Blocks-Debuggers	569
Mit anderen Werkzeug debuggen	571
Standarddebugger	572
Eine Code::Blocks-Anwendung mit Argumenten auf der Befehlszeile debuggen	572
<b>Kapitel 21</b>	
<b>Den Code anhalten und untersuchen</b>	<b>573</b>
Haltepunkte für Fortgeschrittene	574
Haltepunkt in Code::Blocks setzen	575
Haltepunkte aktivieren und deaktivieren	577
Variablen beobachten, untersuchen und ändern	579
Variablen beobachten	580
Objekte beobachten	582
Werte ändern	583
<b>Kapitel 22</b>	
<b>Der Stack</b>	<b>585</b>
Ihre Daten stapeln	585
Im Stack herumreisen	586
Lokale Variablen speichern	589
Mit weitergehenden Programmfunktionen debuggen	591
In Assemblercode einer Spur nachgehen	591

## **Teil V**

### **Dateien lesen und schreiben**

**595**

#### **Kapitel 23**

##### **Informationen mit der Bibliothek »Streams« ablegen**

**597**

Warum Streams notwendig sind	598
Mit der Streams-Bibliothek programmieren	599
Die richtige Headerdatei erhalten	599
Eine Datei öffnen	601
Umgang mit Fehlern beim Öffnen einer Datei	604
Hissen Sie die ios-Flaggen	606

#### **Kapitel 24**

##### **Mit Output-Streams schreiben**

**611**

Mit dem <<-Operator einfügen	611
Formatieren Sie Ihre Ausgabe	613
Mit Flags formatieren	614
Die Präzision festlegen	617
Felder erstellen und ihre Breite setzen	620

#### **Kapitel 25**

##### **Aus Input-Streams lesen**

**625**

Mit Operatoren extrahieren	625
Mit dem Dateiende umgehen	628
Verschiedene Typen einlesen	634
Formatierte Eingabewerte einlesen	637

#### **Kapitel 26**

##### **Mit Verzeichnissen und Dateien arbeiten**

**639**

Mit Verzeichnissen arbeiten	640
Ein Verzeichnis erstellen	640
Ein Verzeichnis löschen	641
Den Inhalt eines Verzeichnisses ermitteln	642
Dateien kopieren	645
Dateien und Verzeichnisse verschieben und umbenennen	647

<b>Kapitel 27</b>	
<b><i>Streamen Sie Ihre eigenen Klassen</i></b>	<b>651</b>
Eine Klasse als Text streamen	651
Einen Stream manipulieren	655
Was ist ein Manipulator?	655
Schreiben Sie Ihren eigenen Manipulator	657
<b>Teil VI</b>	
<b><i>Anwendungen planen und entwerfen</i></b>	<b>663</b>
<b>Kapitel 28</b>	
<b><i>Das Programm mit UML beschreiben</i></b>	<b>665</b>
Zu UML aufsteigen	665
Mit UML modellieren	669
Mit UML schematisiert darstellen und entwerfen	670
Mit UML und dem Rational Unified Process entwickeln	675
Iterativ sprechen	676
Rein in die Phase, raus aus der Phase	678
Die Konzeptionsphase	680
Die Entwurfsphase	681
Die Konstruktionsphase	681
Die Übergabephase	682
Und weiter geht's mit UML	683
<b>Kapitel 29</b>	
<b><i>Die Klasse mit UML strukturieren</i></b>	<b>685</b>
Klassen zeichnen	686
Klassen mit UML abbilden	687
UML und Vererbung	690
Klassen aggregieren und zusammensetzen	691
Komponenten erstellen	693
Die Software verteilen	696
<b>Kapitel 30</b>	
<b><i>Mit UML ein Verhalten aufzeigen</i></b>	<b>699</b>
Objekte zeichnen	699
Anwendungsfälle inspizieren	701
Anwendungsfälle erweitern	704
Anwendungsfälle und Anforderungen in Übereinstimmung bringen	704

Sequenzdiagramme	705
Sequenzdiagramme mit Anmerkungen versehen	708
Schleifen und Vergleiche in Sequenzdiagrammen	708
Kollaborationsdiagramme	711
Aktivitätsdiagramme	713
Zustandsdiagramme	714
 <b>Kapitel 31</b>	
<b>Anwendungen mit UML modellieren</b>	<b>717</b>
UML-Schmankerl	717
Symbole packen	717
Diagramme mit Anmerkungen versehen	721
Symbole kennzeichnen	721
Frei zu sein bedarf es wenig	722
C++ und UML	724
Aufzählungen zeichnen	724
Statische Mitglieder einbinden	725
Klassen über Templates mit Parametern versorgen	726
 <b>Teil VII</b>	
<b>Fortgeschrittenes C++</b>	<b>729</b>
 <b>Kapitel 32</b>	
<b>Eine Reise durch die Standardbibliothek</b>	<b>731</b>
Kategorien der Standardbibliothek	732
Container	733
Iteratoren	733
Algorithmen	734
Funktionen	735
Utilities	737
Allokatoren	737
Polymorphe Allokatoren	738
Strings mit einem Hash parsen	738
Informationen über einen Iterator mit wahlfreiem Zugriff abrufen	741
Werte mit dem Find-Algorithmus ermitteln	744
Den Zufallszahlengenerator verwenden	746
Mit min und max arbeiten	748

## **Kapitel 33**

### **Eigene Templates erstellen**

**749**

Ein einfaches Mathematik-Template erstellen	749
Ein Struktur-Template erstellen	751
Ein Klassen-Template entwickeln	754
Template-Spezialisierung	757
Eine Bibliothek erstellen	759
Das Bibliotheksprojekt definieren	759
Das Bibliotheksprojekt konfigurieren	762
Schreiben Sie den Code für die Bibliothek	764
Setzen Sie Ihre Bibliothek ein	765

## **Kapitel 34**

### **Boost erforschen**

**767**

Boost verstehen	768
Features von Boost	768
Lizenzen	769
Bezahlter Support	770
Boost herunterladen und für Code::Blocks installieren	770
Packen Sie Boost aus	770
Setzen Sie die Header-Only-Bibliotheken ein	772
Bauen Sie die Bibliotheken	772
Testen Sie die Installation	774
Bauen Sie Ihre erste Boost-Anwendung mit DateTime	779
Die Boost-Tools erstellen	782
Boost.Build einsetzen	784
Einen erfolgreichen Build durchführen	785
Die Beispiele einsetzen	785
Regression verwenden	786
Inspect einsetzen	786
BoostBook verstehen	789
QuickBook einsetzen	790
»bcp« einsetzen	791
Wave einsetzen	792

## **Kapitel 35**

### **Mit Boost weitergehen**

**793**

Strings mit RegEx durchsuchen	794
Die RegEx-Bibliothek hinzufügen	794
Den RegEx-Code schreiben	798
Strings mit dem Tokenizer aufteilen	800
Numerische Umwandlungen vornehmen	801
Bessere Schleifen mit Foreach schreiben	804
Mit Filesystem auf das Betriebssystem zugreifen	807

## **Anhang**

### ***Automatisieren Sie Ihre Anwendungen durch Makefiles***

**811**

Kompilieren und Linken	811
Automatisieren Sie Ihre Arbeit	813
Mit Inferenz-Regeln Anwendungen bauen	814
Regeln verwenden, die von anderen Regeln abhängen	816
Bestimmte Elemente erstellen	816
Von mehreren Dateien abhängig sein	817
Kompilieren und Linken mit make	818
Aufräumen	819
Makros einsetzen	819
Das Beste aus Makefiles herausholen	820

### ***Stichwortverzeichnis***

**821**