

Inhaltsverzeichnis

| | |
|---|-----------|
| Über den Autor | 9 |
| Widmung | 10 |
| Einführung | 23 |
| Über dieses Buch | 23 |
| Törichte Annahmen über den Leser | 24 |
| Wie dieses Buch organisiert ist | 25 |
| Teil I: Programmieren in C++ – die ersten Schritte | 25 |
| Teil II: Ein Programm schreiben: Entscheidungen, Entscheidungen | 25 |
| Teil III: Prozedural programmieren | 25 |
| Teil IV: Datenstrukturen | 25 |
| Teil V: Objektorientierte Programmierung | 26 |
| Teil VI: Für Fortgeschrittene | 26 |
| Teil VII: Der Top-Ten-Teil | 26 |
| Symbole in diesem Buch | 26 |
| Wie es weitergeht | 27 |
| | |
| Teil I | |
| Programmieren in C++ – die ersten Schritte | 29 |
| Kapitel 1 | |
| Was ist ein Programm? | 31 |
| Worin unterscheidet sich mein Sohn von einem Computer? | 31 |
| Einen »menschlichen Computer« programmieren | 33 |
| Den Algorithmus erstellen | 33 |
| Die Entwicklung der Reifenwechsel-Sprache | 34 |
| Das Programm erstellen | 34 |
| Computerprozessoren | 38 |
| Computersprachen | 38 |
| Höhere Sprachen | 40 |
| Die Sprache C++ | 40 |
| | |
| Kapitel 2 | |
| Code::Blocks installieren | 43 |
| Der Kompilierungsvorgang | 43 |
| Code::Blocks installieren | 45 |
| Installation unter Windows | 45 |
| Installation unter Ubuntu Linux | 48 |
| Installation unter Mac OS | 50 |
| Code::Blocks einrichten | 54 |

| | |
|--------------------------------------|----|
| Die Code::Blocks-Installation testen | 57 |
| Das Projekt anlegen | 57 |
| Ihr Standardprojekt testen | 62 |

Kapitel 3

Ihr erstes Programm **65**

| | |
|-------------------------------|----|
| Ein neues Projekt anlegen | 65 |
| Dateinamenerweiterungen | 67 |
| Die Eingabe Ihres Programms | 69 |
| Das Programm erstellen | 71 |
| Was alles schiefgehen kann | 71 |
| Falsch geschriebene Befehle | 71 |
| Fehlendes Semikolon | 73 |
| Die Beispieldateien verwenden | 74 |
| Das Programm ausführen | 75 |
| Wie das Programm funktioniert | 75 |
| Die Vorlage | 76 |
| Das Conversion-Programm | 77 |

Teil II

Ein Programm schreiben: Entscheidungen, Entscheidungen **79**

Kapitel 4

Integer-Ausdrücke – für ganze Zahlen **81**

| | |
|--|----|
| Variablen deklarieren | 81 |
| Variablennamen | 82 |
| Einer Variablen einen Wert zuweisen | 83 |
| Eine Variable bei der Deklaration initialisieren | 83 |
| Ganzzahlige Konstanten | 84 |
| Ausdrücke | 85 |
| Binäre Operatoren | 86 |
| Zusammengesetzte Ausdrücke auflösen | 87 |
| Unäre Operatoren | 88 |
| Die speziellen Zuweisungsoperatoren | 90 |

Kapitel 5

Zeichenausdrücke **93**

| | |
|-----------------------------------|----|
| Character-Variablen definieren | 93 |
| Zeichen codieren | 93 |
| Beispiel für die Zeichencodierung | 96 |
| Zeichenketten codieren | 98 |
| Sonderzeichen-Konstanten | 98 |

Kapitel 6**Entscheidungen, Entscheidungen! 101**

| | |
|--------------------------------------|-----|
| Die if-Anweisung | 101 |
| Vergleichsoperatoren | 102 |
| Geschweifte Klammern sind kein Luxus | 104 |
| Und andernfalls? | 106 |
| Verschachtelte if-Anweisungen | 108 |
| Zusammengesetzte bedingte Ausdrücke | 111 |

Kapitel 7**Ausführungspfade wechseln 113**

| | |
|--|-----|
| Mit der switch-Anweisung den Programmablauf steuern | 113 |
| Durchgerasselt: Habe ich es kaputtgemacht? | 116 |
| Implementierung eines einfachen Taschenrechners mit der switch-Anweisung | 117 |

Kapitel 8**Programme debuggen, Teil I 121**

| | |
|---|-----|
| Fehlertypen identifizieren | 121 |
| Fehler vermeiden | 122 |
| Codieren mit Stil | 122 |
| Namenskonventionen für Variablen festlegen | 123 |
| Den ersten Fehler finden – mit ein bisschen Unterstützung | 124 |
| Den Laufzeitfehler finden | 125 |
| Testdaten formulieren | 125 |
| Tests durchführen | 126 |
| Sehen wir nach, was das Programm macht | 126 |

Teil III**Prozedural programmieren 129****Kapitel 9****while – ein ewiger Kreislauf 131**

| | |
|-------------------------------|-----|
| while-Schleifen | 131 |
| Aus einer Schleife ausbrechen | 134 |
| Verschachtelte Schleifen | 137 |

Kapitel 10**Weiter mit for-Schleifen 143**

| | |
|---------------------------------------|-----|
| Die vier Teile jeder Schleife | 143 |
| Wir betrachten ein Beispiel | 145 |
| Mit dem Komma-Operator mehr erledigen | 146 |

Kapitel 11**Funktionen** **151**

| | |
|--|-----|
| Aufgaben in Funktionen zerlegen | 151 |
| Die Arbeitsweise von Funktionen verstehen | 152 |
| Eine Funktion schreiben und verwenden | 153 |
| Dinge zurückgeben | 154 |
| Ein Beispiel | 155 |
| An Funktionen Argumente übergeben | 158 |
| Funktion mit Argumenten | 158 |
| Funktionen mit mehreren Argumenten | 160 |
| main() | 161 |
| Funktionsprototyp-Deklarationen definieren | 161 |

Kapitel 12**Programme in Module unterteilen** **165**

| | |
|---------------------------------------|-----|
| Programme aufsplitten | 165 |
| Teilen ist gar nicht so schwer | 166 |
| Factorial.cpp erstellen | 166 |
| Eine #include-Datei erstellen | 169 |
| #include-Dateien einbinden | 171 |
| main.cpp erstellen | 172 |
| Das Ergebnis erstellen | 174 |
| Verwendung der C++-Standardbibliothek | 174 |
| Gültigkeitsbereiche von Variablen | 175 |

Kapitel 13**Programme debuggen, Teil 2** **177**

| | |
|--|-----|
| Debugging eines Programms mit Rechenschwäche | 177 |
| Einheitentests durchführen | 179 |
| Eine Funktion für Tests ausstatten | 180 |
| Zurück zum Einheitentest | 184 |

Teil IV**Datenstrukturen** **187****Kapitel 14****Andere numerische Variablentypen** **189**

| | |
|---------------------------------|-----|
| Die Grenzen von Integern in C++ | 189 |
| Integer-Abrundung | 189 |
| Begrenzter Wertebereich | 190 |

| | |
|--|------------|
| Ein »doppelter« Typ für reelle Zahlen | 191 |
| Eine Lösung für das Abrundungsproblem | 191 |
| Wenn ein Integer kein Integer ist | 192 |
| Die Grenzen eines double erkennen | 193 |
| Variablengröße – die lange und die kurze Form | 195 |
| Wie weit reichen Zahlen? | 197 |
| Konstantentypen | 198 |
| Funktionen unterschiedliche Typen übergeben | 199 |
| Funktionsnamen überladen | 199 |
| Gemischtes Überladen | 200 |
| | |
| Kapitel 15 | |
| Arrays | 203 |
| Was ist ein Array? | 203 |
| Ein Array deklarieren | 204 |
| Array-Elemente über einen Index ansprechen | 204 |
| Ein Beispiel | 206 |
| Ein Array initialisieren | 208 |
| | |
| Kapitel 16 | |
| Arrays mit Charakter | 211 |
| Das ASCII-Zero-Character-Array | 211 |
| Ein ASCII-Z-Array deklarieren und initialisieren | 212 |
| Ein Beispiel | 213 |
| Ein detaillierteres Beispiel | 215 |
| Hackerabwehr | 218 |
| Muss ich das wirklich alles machen? | 220 |
| | |
| Kapitel 17 | |
| Zeiger in C++ | 223 |
| Was ist ein Zeiger? | 223 |
| Einen Zeiger deklarieren | 224 |
| Einer Funktion Argumente übergeben | 226 |
| Argumente als Wert übergeben | 226 |
| Argumente als Referenz übergeben | 229 |
| Das große Ganze | 231 |
| Typen von Referenzargumenten | 233 |
| Speicherstapel | 233 |
| Brauchen Sie wirklich ein neues Schlüsselwort? | 234 |
| Vergessen Sie nicht, zum Schluss wieder aufzuräumen! | 235 |
| Ein Beispiel | 236 |

Kapitel 18

C++-Zeiger – auf den zweiten Blick **239**

| | |
|---|-----|
| Zeiger und Arrays | 239 |
| Operationen für Zeiger | 239 |
| Zeiger-Addition im Vergleich zur Indizierung eines Arrays | 242 |
| Der Inkrementoperator für Zeiger | 244 |
| Warum plagen wir uns mit Array-Zeigern? | 247 |
| Operationen für verschiedene Zeigertypen | 248 |
| Die Sache mit den Konstanten | 248 |
| Unterschiede zwischen Zeigern und Arrays | 249 |
| Meine Argumente von main() | 250 |
| Zeigerarrays | 250 |
| Arrays mit Argumenten | 251 |

Kapitel 19

Programmieren mit Klasse **259**

| | |
|---------------------|-----|
| Daten gruppieren | 259 |
| Die Klasse | 260 |
| Das Objekt | 261 |
| Arrays von Objekten | 262 |
| Ein Beispiel | 263 |

Kapitel 20

Programme debuggen, Teil 3 **271**

| | |
|--|-----|
| Ein neuer Ansatz für das Debugging | 271 |
| Die Lösung | 272 |
| Debuggen Schritt für Schritt | 272 |
| Den Debugger starten | 275 |
| Navigation durch ein Programm mit dem Debugger | 278 |
| Den (ersten) Fehler korrigieren | 282 |
| Den zweiten Fehler finden und korrigieren | 284 |

Teil V

Objektorientierte Programmierung **287**

Kapitel 21

Was ist objektorientierte Programmierung? **289**

| | |
|-------------------------------------|-----|
| Abstraktion und Mikrowellenöfen | 289 |
| Prozedurale Nachos | 290 |
| Objektorientierte Nachos | 291 |
| Klassifizierung und Mikrowellenöfen | 291 |

| | |
|---|------------|
| Warum sollten wir Objekte auf diese Weise aufbauen? | 292 |
| Abgeschlossene Klassen | 293 |
| Kapitel 22 | |
| <i>Strukturiertes Spiel: Wie Klassen Dinge erledigen</i> | 295 |
| Unsere Objekte aktivieren | 295 |
| Eine Elementfunktion erstellen | 296 |
| Eine Elementfunktion definieren | 297 |
| Namen für Klassenelemente | 298 |
| Aufruf einer Elementfunktion | 299 |
| Zugriff auf andere Elemente von einer Elementfunktion aus | 300 |
| Eine Elementfunktion hinter der Klasse halten | 301 |
| Elementfunktionen überladen | 302 |
| Kapitel 23 | |
| <i>Zeiger auf Objekte</i> | 305 |
| Zeiger auf Objekte | 305 |
| Pfeilsyntax | 306 |
| Aufruf aller Elementfunktionen | 306 |
| Funktionen Objekte übergeben | 307 |
| Aufruf einer Funktion mit einem Objektwert | 307 |
| Aufruf einer Funktion mit einem Objektzeiger | 309 |
| Ein Beispiel | 310 |
| Objekte auf dem Stapel reservieren | 314 |
| Kapitel 24 | |
| <i>Bitte nicht stören: Geschützte Elemente</i> | 317 |
| Elemente schützen | 317 |
| Warum Sie geschützte Elemente brauchen | 317 |
| Elemente schützen | 318 |
| Also? | 321 |
| Ein Freund, ein guter Freund ... | 322 |
| Kapitel 25 | |
| <i>Objekten einen guten Start verschaffen</i> | 325 |
| Der Konstruktor | 325 |
| Einschränkungen von Konstruktoren | 327 |
| Kann ich ein Beispiel sehen? | 327 |
| Datenelemente konstruieren | 330 |
| Destruktoren | 333 |
| Ein Beispiel | 334 |
| Datenelemente zerstören | 337 |

Kapitel 26

Konstruktive Argumente **341**

| | |
|--|-----|
| Konstruktoren mit Argumenten | 341 |
| Ein Beispiel | 342 |
| Den Konstruktor überladen | 346 |
| Der Standard-Standardkonstruktor | 350 |
| Datenelemente konstruieren | 352 |
| Datenelemente mit dem Standardkonstruktor initialisieren | 352 |
| Datenelemente mit einem anderen Konstruktor initialisieren | 354 |
| Ein Beispiel | 357 |
| Neu in C++ 2011 | 360 |

Kapitel 27

Kopieren mit dem Copy-Konstruktor **361**

| | |
|----------------------------------|-----|
| Ein Objekt kopieren | 361 |
| Der Standard-Copy-Konstruktor | 362 |
| Ein Beispiel | 363 |
| Einen Copy-Konstruktor erstellen | 366 |
| Kopien vermeiden | 369 |

Teil VI

Für Fortgeschrittene **371**

Kapitel 28

Eine Klasse vererben **373**

| | |
|---------------------------------|-----|
| Vorteile der Vererbung | 373 |
| Die Sprachbesonderheiten | 374 |
| Vererbung in C++ implementieren | 375 |
| Ein Beispiel | 376 |
| Eine HAT_EIN-Beziehung | 380 |

Kapitel 29

Virtuelle Funktionen – Realität? **383**

| | |
|---------------------------------|-----|
| Elementfunktionen überschreiben | 383 |
| Frühes Binden | 384 |
| Mehrdeutiger Fall | 386 |
| Eine späte Bindung eingehen | 388 |
| Wann nicht virtuell? | 391 |
| Virtuelle Aspekte | 393 |

Kapitel 30

| | |
|--|------------|
| Zuweisungsoperatoren überladen | 397 |
| Einen Operator überladen | 397 |
| Das Überladen des Zuweisungsoperators ist kritisch | 398 |
| Ein Beispiel | 400 |
| Ihren Eigenen schreiben (oder nicht) | 403 |

Kapitel 31

| | |
|---|------------|
| Stream-I/O | 405 |
| Wie Stream-I/O funktioniert | 405 |
| Stream-Eingabe/Ausgabe | 407 |
| Ein Eingabe-Objekt erstellen | 407 |
| Ein Ausgabe-Objekt erstellen | 408 |
| Öffnen-Modi | 409 |
| Was ist der Binärmodus? | 410 |
| Und in welchem Status ist eine Datei? | 410 |
| Weitere Elementfunktionen der fstream-Klassen | 415 |
| Streams direkt lesen und schreiben | 415 |
| Formatsteuerung | 419 |
| Und was macht eigentlich endl? | 422 |
| Manipulatoren manipulieren | 422 |
| Die stringstream-Klassen | 423 |

Kapitel 32

| | |
|---|------------|
| Machen wir eine Ausnahme! | 429 |
| Der Ausnahmemechanismus | 429 |
| Betrachten wir den Ausnahmemechanismus im Detail! | 432 |
| Spezielle Aspekte für das Aufwerfen von Ausnahmen | 433 |
| Eine benutzerdefinierte Ausnahmeklasse erstellen | 433 |
| Einschränkungen von Ausnahmeklassen | 437 |

Teil VII

| | |
|-------------------------|------------|
| Der Top-Ten-Teil | 439 |
|-------------------------|------------|

Kapitel 33

| | |
|---|------------|
| Zehn Methoden, Fehler zu vermeiden | 441 |
| Aktivieren Sie alle Warnungen und Fehlermeldungen! | 441 |
| Gewöhnen Sie sich einen klaren und konsistenten Programmierstil an! | 442 |
| Kommentieren Sie den Code, während Sie ihn schreiben! | 442 |
| Durchlaufen Sie jeden Pfad mindestens einmal im Einzelschrittmodus im Debugger! | 443 |

| | |
|--|------------|
| Begrenzen Sie die Sichtbarkeit! | 443 |
| Verwalten Sie Ihren Stapel! | 444 |
| Überschreiben Sie Zeiger mit 0, nachdem Sie gelöscht haben, worauf sie zeigen! | 444 |
| Verarbeiten Sie Fehler mit Ausnahmen! | 445 |
| Deklarieren Sie Destruktoren als virtuell! | 445 |
| Stellen Sie einen Copy-Konstruktor und einen überladenen Zuweisungsoperator bereit! | 445 |
| | |
| Kapitel 34 | |
| Zehn Dinge, die in diesem Buch nicht abgedeckt sind | 447 |
| Der goto-Befehl | 447 |
| Der ternäre Operator | 448 |
| Binäre Logik | 448 |
| Aufzählungstypen | 448 |
| Namensräume | 449 |
| Rein virtuelle Funktionen | 449 |
| Die string-Klasse | 450 |
| Mehrfachvererbung | 450 |
| Templates und die Standard Template Library | 450 |
| Lambda-Funktionen | 451 |
| | |
| Stichwortverzeichnis | 453 |