Inhalt

1 Einstieg in die Welt von C++ 17

1.1 Der C++-Standard 17
1.2 Die nötigen Werkzeuge für C++ 18
1.3 Übersetzen mit g++ und clang++ 22
1.4 Übersetzen mit einer Entwicklungsumgebung 24
1.5 Listings zum Buch 28
1.6 Kontrollfragen und Aufgaben im Buch 28
1.7 Aufgabe 28

2 Erste Schritte in C++ 30

2.1 Das erste Programm in C++ 30
2.2 Anweisungen und Ausdrücke 32
2.3 Die Standard-Eingabe- und -Ausgabestreams 33
2.3.1 Die Streams von C++ 34
2.3.2 Ausgabe mit »std::cout« 34
2.3.3 Eingabe mit »std::cin« 35
2.3.4 Ausgabe mit »std::cerr« 36
2.4 Einige Begriffe zu C++ 38
2.4.1 Bezeichner 38
2.4.2 Literale 39
2.4.3 Kommentare 39
2.5 Kontrollfragen und Aufgaben 40

Bibliografische Informationen
http://d-nb.info/1241344868

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

3 Die eingebauten C++-Basisdatentypen 41

3.1 Variablen ... 41

3.2 Definition und Deklaration von Variablen ... 42

3.3 Initialisierung und Zuweisung von Werten ... 43

3.4 Ganzzahltypen ... 45

3.4.1 Literale von Ganzzahltypen ... 50

3.5 Ganzzahldatentyp für Zeichen ... 53

3.5.1 Weitere Datentypen für Zeichen ... 58

3.5.2 Die Unicode-Typen »char8_t«, »char16_t« und »char32_t« ... 60

3.6 Fließkommazahlentypen ... 62

3.6.1 Literale von Fließkommazahlen ... 64

3.7 Der »auto«-Typ ... 64

3.8 Konstanten ... 66

3.9 Die Byte-Größe mit dem »sizeof«-Operator ... 67

3.10 Limits für die Basisdatentypen ... 68

3.11 Kontrollfragen und Aufgaben ... 71

4 Arbeiten mit den eingebauten Typen 72

4.1 Arithmetische Operatoren ... 72

4.1.1 Kurzschreibweise arithmetischer Operatoren ... 76

4.1.2 Inkrement- und Dekrementoperator ... 77

4.2 Ungenaue Fließkommazahlen ... 79

4.3 Typumwandlung ... 81

4.3.1 Implizite Umwandlung durch den Compiler ... 81

4.3.2 Automatische Typumwandlung beschränken 84
4.3.3 Explizite Typumwandlung 85
4.4 Kontrollfragen und Aufgaben 87

5 Kontrollstrukturen 88

5.1 Der eingebaute Datentyp »bool« 88
5.2 Vergleichsoperatoren 89
5.3 Bedingte Anweisung mit »if« 91
5.4 Anweisungsblock für Kontrollstrukturen 93
5.5 Alternative »else«-Verzweigung 94
5.6 Bedingte Anweisung mit Initialisierung 95
5.7 Mehrfache Verzweigung 96
5.8 Der Bedingungsoperator »?:« 99
5.9 Logische Operatoren 100
5.10 Die Fallunterscheidung – »switch« 102
5.11 Die kopfgesteuerte »while«-Schleife 105
5.12 Die fußgesteuerte »do while«-Schleife 107
5.13 Die Zählschleife »for« 108
5.14 Kontrollierte Sprunganweisungen 112
5.14.1 Die »break«-Anweisung 112
5.14.2 Die »continue«-Anweisung 113
5.15 Kontrollfragen und Aufgaben 115

6 Arrays und Strings 116

6.1 Arrays ... 116
6.1.1 Der C++-Container »std::vector« ... 118
6.1.2 Der C++-Container »std::array« ... 125
6.1.3 C-Arrays ... 126
6.2 Strings in C++ ... 130
6.2.1 Der C++-Container »std::string« ... 130
6.2.2 Unterstützung von Unicode ... 132
6.2.3 C-Zeichenketten ... 133
6.2.4 Zeichenkettenliterale ... 135
6.3 Kontrollfragen und Aufgaben ... 136

7 Referenzen und Zeiger 137

7.1 Referenzen ... 137
7.2 Zeiger ... 139
7.2.1 Die Syntax von Zeigern ... 140
7.2.2 Zeiger dereferenzieren ... 142
7.2.3 Der Zeiger »nullptr« ... 144
7.2.4 Zeiger prüfen ... 145
7.2.5 Adresse einer Referenz ... 146
7.2.6 Verwendung von Zeigern und Alternativen ... 147
7.3 Kontrollfragen und Aufgaben ... 148

8 Funktionen 150

8.1 Grundlage zu den Funktionen ... 150
8.1.1 Funktionen definieren ... 151
8.1.2 Funktionen aufrufen ... 152

8.1.3 Funktionen deklarieren ... 153
8.1.4 Funktionsparameter (Call-by-Value) ... 155
8.1.5 Konstante Funktionsparameter ... 157
8.1.6 Standardparameter ... 158
8.1.7 Rückgabe aus Funktionen ... 160
8.1.8 Funktionen überladen ... 163
8.1.9 Gültigkeitsbereich und Sichtbarkeit von Variablen ... 166
8.1.10 Die »main()«-Funktion ... 169
8.1.11 Aufruf eines Programms mit Parametern ... 169
8.1.12 Programmende ... 171
8.2 Referenzen als Parameter und Rückgabe ... 171
8.2.1 Referenzen als Parameter ... 173
8.2.2 Konstante Funktionsparameter ... 174
8.2.3 Referenzen als Rückgabe ... 176
8.3 Zeiger als Parameter und Rückgabewert ... 178
8.3.1 Referenzen vs. Zeiger als Parameter ... 179
8.4 Übergabe großer Elemente als Funktionsparameter ... 179
8.5 C-Arrays oder C-Strings als Funktionsparameter ... 181
8.6 Kontrollfragen und Aufgaben ... 182

9 Modularisierung und Präprozessor 184

9.1 Präprozessor-Direktiven ... 184
9.1.1 Die »#include«-Direktive ... 185
9.1.2 Die »#define«-Direktive ... 186
9.1.3 Bedingte Kompilierung ... 187
9.2 Modularisierung ... 189
9.2.1 Aufteilung ... 189
9.2.2 Die öffentliche Schnittstelle (Headerdatei) ... 191

9.2.3 Die nicht öffentliche(n) Datei(en) ... 192
9.2.4 Die Client-Datei ... 193
9.2.5 Aufgabe ... 194
9.2.6 Nur Objektcode oder Bibliothek vorhanden ... 195
9.3 Namensräume ... 195
9.3.1 Namensraum deklarieren und verwenden ... 196
9.3.2 Namensraum verschachteln ... 199
9.3.3 Ein Namensraum ist ein eigener Gültigkeitsbereich 199
9.3.4 Namensraum mit »using« importieren ... 202
9.3.5 Einzelne Bezeichner mit »using« importieren ... 203
9.3.6 Aliasse für Namensräume ... 203
9.3.7 Anonymer Namensraum ... 204
9.3.8 Der Namensraum »std« ... 205
9.4 Spezifizierer und Qualifikatoren ... 206
9.4.1 Das Schlüsselwort »static« ... 207
9.4.2 Das Schlüsselwort »extern« ... 209
9.4.3 Das Schlüsselwort »constexpr« ... 210
9.4.4 Das Schlüsselwort »const« ... 211
9.4.5 Das Schlüsselwort »inline« ... 212
9.5 Kontrollfragen und Aufgaben ... 213

10 Strukturen, Aufzählungen und dynamische Speicherobjekte 215

10.1 Erste eigene Datentypen mit Strukturen ... 215
10.1.1 Strukturen definieren, Elemente erzeugen und initialisieren ... 216
10.1.2 Zugriff auf die Strukturelemente ... 218
10.1.3 Zugriff auf die Elemente in einer Funktion ... 219
10.1.4 Strukturen in einem Vektor oder Array ... 220

10.1.5 Methoden statt Funktionen 221
10.1.6 Strukturen vergleichen 223
10.2 Aufzählungstyp »enum« 223
10.3 Eigene Namen mit »using« 225
10.4 Dynamische Speicherobjekte 226
10.4.1 Dynamisch Objekte mit »new« und »delete« anlegen und freigeben 228
10.4.2 Dynamisch Arrays mit »new[ ]« anlegen und mit »delete[ ]« freigeben 231
10.4.3 Der smarte »unique_ptr«-Pointer 233
10.5 Kontrollfragen und Aufgaben 236

11 Klassen 237

11.1 Klassen 237
11.1.1 Klassendefinition 237
11.1.2 Zugriffskontrolle mit »public« und »private« 239
11.1.3 Methoden definieren 243
11.1.4 Objekte erzeugen und benutzen 245
11.2 Konstruktoren 251
11.2.1 Konstruktoren deklarieren 252
11.2.2 Konstruktoren definieren 253
11.2.3 Konstruktoren delegieren 256
11.2.4 Der Standardkonstruktor (Default-Konstruktor) 258
11.2.5 Implizite Konvertierungen und ihre Verhinderung – »explicit« 260
11.2.6 Der Kopierkonstruktor (Copy-Konstruktor) 262
11.2.7 Der Verschiebekonstruktor (Move-Konstruktor) 264
11.3 Destruktoren 268
11.3.1 Die Lebensdauer eines Objekts 268

11.3.2 Wann ist ein Destruktor erforderlich? ... 268
11.3.3 Destruktor deklarieren ... 269
11.3.4 Destruktor definieren ... 270
11.4 Weitere Formen von Methoden ... 273
11.4.1 »inline«-Methoden ... 273
11.4.2 Konstante Methoden (»nur-lesend«) ... 276
11.4.3 Konstante Methoden explizit ausschließen ... 278
11.4.4 »this«-Zeiger ... 279
11.5 Kontrollfragen und Aufgaben ... 281

12 Objekte und Klassenelemente 283

12.1 Objekt als Parameter ... 283
12.1.1 Objekte an eine Funktion übergeben ... 283
12.1.2 Objekte an eine Methode übergeben ... 286
12.2 Freundfunktionen (»friend«) ... 288
12.3 Objekte einer Klasse als Rückgabewerte ... 290
12.3.1 Referenzen auf eine Klasse als Rückgabewerte ... 293
12.4 Arrays von Objekten ... 296
12.5 Dynamische Objekte ... 297
12.6 Klassenobjekte als Klassenattribute ... 299
12.7 Containerklasse als Klassenattribut ... 304
12.8 Smart Pointer als Klassenattribut ... 307
12.9 Statische und konstante Klassenelemente ... 311
12.9.1 Statische Klassenelemente ... 311
12.9.2 Konstante Klassenelemente ... 316
12.9.3 Rohe Zeiger als Klassenelemente oder direkt die Nullregel ... 317

12.10 Die Nullregel (Rule of Zero) 318
12.10.1 Die großen fünf 319
12.11 Kontrollfragen und Aufgaben 322

13 Operatoren überladen 323

13.1 Das Schlüsselwort »operator« 325
13.2 Zweistellige (arithmetische) Operatoren überladen 326
13.2.1 Operatorüberladung als Methode einer Klasse 328
13.2.2 Operatorüberladung als globale Hilfsfunktion 331
13.3 Einstellige Operatoren überladen 333
13.4 Den Zuweisungsoperator überladen 337
13.5 Ausgabe- und Eingabeoperatoren überladen 342
13.5.1 Den Ausgabeoperator »<<« überladen 342
13.5.2 Den Eingabeoperator »>>« überladen 343
13.6 Vergleichsoperatoren 345
13.6.1 Der Drei-Wege-Vergleichsoperator 346
13.7 Weitere Operatorüberladungen 348
13.8 Konvertierungsoperatoren 348
13.8.1 Der Konvertierungskonstruktor 349
13.8.2 Die Konvertierungsfunktion 350
13.9 Kontrollfragen und Aufgaben 352

14 Vererbung (Abgeleitete Klassen) 354

14.1 Die Vorbereitung 355
14.2 Das Ableiten einer Klasse 357
14.2.1 Erben und erweitern 358

14.2.2 »public«-Zugriffsrechte einer abgeleiteten Klasse ... 359
14.2.3 Methoden überschreiben ... 361
14.2.4 Konstruktoren ... 362
14.2.5 Programmbeispiel ... 364
14.2.6 Konstruktoren vererben ... 365
14.2.7 Destruktor ... 367
14.2.8 Die Zugriffsspezifikation »protected« ... 367
14.2.9 Implizite Typumwandlung abgeleiteter Klassen ... 369
14.2.10 Überschreiben mit virtuellen Methoden ... 370
14.2.11 Abstrakte Klassen und rein virtuelle Methoden ... 373
14.3 Kontrollfragen und Aufgaben ... 375

15 Templates 376

15.1 Funktions-Templates ... 376
15.1.1 Funktions-Template definieren ... 377
15.1.2 Funktions-Templates über mehrere Module ... 380
15.1.3 Ein Funktions-Template spezialisieren ... 380
15.1.4 Templates mit verschiedenen Parametern ... 382
15.1.5 Explizite Template-Argumente ... 383
15.1.6 Wiederverwendbare Templates ... 384
15.2 Klassen-Templates ... 385
15.2.1 Klassen-Template definieren ... 385
15.2.2 Template-Methoden definieren ... 386
15.2.3 Template-Methoden spezialisieren ... 387
15.2.4 Klassen-Template instanziieren ... 388
15.2.5 Klassen-Template mit mehreren formalen Parametern ... 390

15.3 Templates der Standardbibliothek 390
15.3.1 Iteratoren 391
15.3.2 Algorithmen 394
15.4 Kontrollfragen und Aufgaben 397

16 Ausnahmebehandlung (Fehlerbehandlung) 399

16.1 Eine Ausnahme auslösen 401
16.2 Ausnahme auffangen und behandeln 402
16.2.1 Alternatives »catch (...)« 405
16.2.2 Ausnahmen aus der Standardbibliothek 407
16.2.3 Ausnahme mit »throw« weiterwerfen 409
16.2.4 »noexcept« 411
16.2.5 Stack-Abwicklung 412
16.3 Ausnahmeklassen (Fehlerklassen) 412
16.4 Standardausnahmen 415
16.4.1 Die virtuelle Methode »what()« 416
16.4.2 Die Standardausnahmen verwenden 416
16.5 Systemausnahmen 420
16.6 Kontrollfragen 421

17 Ein-/Ausgabestreams für Dateien 422

17.1 Der Umgang mit Dateien in C++ 422
17.2 Verschiedene Streams für Dateien 422
17.3 Datei öffnen und schließen 423
17.3.1 Verschiedene Modi zum Öffnen von Dateien 425
17.3.2 Byteweise lesen und schreiben 428

17.3.3 Zeilenweise lesen und schreiben 430
17.3.4 Blockweise lesen und schreiben 431
17.3.5 Die Lese- oder Schreibposition ändern 433
17.4 Kontrollfragen und Aufgaben 434

18 Weitere Sprachelemente und die Bibliotheken 435

18.1 Weitere Sprachelemente 435
18.1.1 »decltype« 435
18.1.2 Rückgabesyntax mit nachlaufendem Rückgabetyp 436
18.1.3 Automatische Rückgabetypermittlung 438
18.1.4 Lambda-Funktionen 439
18.2 Smart Pointer 442
18.2.1 Die Move-Semantik 450
18.3 Elemente der Standardbibliothek 452
18.3.1 Sequenzielle Container 453
18.3.2 Assoziative Container 455
18.4 Die Zeitbibliothek 458
18.4.1 Zeitdauer (»duration«) 459
18.4.2 Vorhandene Zeitgeber 460
18.5 Die »ratio«-Bibliothek 463
18.6 Multithreading 464

Lösungen der Übungsaufgaben 466
Index 489