Inhaltsverzeichnis

1 Einführung 1

2 Die Sprache C++ 5
2.1 Geschichte und Paradigmenwandel 5
2.2 Grundlagen 9
2.2.1 Bezeichner in C++ 12
2.2.2 Der Präprozessor 13
2.2.3 Variablen 20
2.2.4 Standarddatentypen 22
2.2.5 Literalkonstanten 25
2.2.6 Konstanten 26
2.2.7 Aufzählungen 27
2.2.8 Arrays 28
2.2.9 Zeiger 29
2.2.10 Referenzen 36
2.2.11 Typenkonvertierung 37
2.2.12 Ausdrücke 38
2.2.13 Operatoren 40
2.2.14 Anweisungen 43
2.2.15 Kontrollstrukturen 44
2.2.16 Funktionen 48
2.2.17 Funktionsüberladung 52
2.2.18 Funktions-Inlining 54
2.2.19 Makros 56
2.2.20 Dynamische Speicherallokation 57
2.2.21 Strukturen und Klassen 64
2.2.22 Typenkonvertierung mit Konstruktoren 88
2.2.23 Globale, automatische und dynamische Instanziierung 91
2.2.24 Speicherklassen 96
2.2.25 Der Scope-Operator 102

2.2.26 Verschachtelte Typen ... 105
2.2.27 Die `friend`-Deklaration ... 107
2.2.28 Statische Methoden und Attribute ... 108
2.2.29 Vererbung ... 109
2.2.30 Virtuelle Methoden und Polymorphismus ... 117
2.2.31 Operatoren der Typenkonvertierung ... 122
2.2.32 Mehrfachvererbung ... 129
2.2.33 Virtuelle Vererbung ... 132
2.2.34 Das Schlüsselwort `const` ... 136
2.2.35 Operatorüberladung ... 139
2.2.36 Exception Handling ... 149

3 Die objektorientierte Programmierung mit C++ ... 165
3.1 Der Klassenbegriff ... 166
3.2 Die Rolle von Patterns und Idiomen ... 168
3.2.1 Der Iterator ... 170
3.2.2 Das Zustandsmuster ... 171
3.2.3 Das Singleton-Muster ... 178
3.3 Datenstrukturen und Containerklassen ... 188
3.3.1 Die Liste ... 189
3.3.2 Der Vektor ... 198
3.3.3 Dynamische Container und das Problem der Speicherfragmentierung ... 200
3.3.4 Verfeinerung des Zugriffs durch überladene Operatoren ... 201
3.4 Arbeiten mit Invarianten ... 202

4 Generische und generative Programmierung mit C++ ... 211
4.1 Templates ... 212
4.1.1 Funktionstemplates ... 213
4.1.2 Klassentemplates ... 216
4.1.3 Methodentemplates ... 217
4.1.4 Instanziierung von Templates ... 218
4.1.5 Member-Templates ... 219
4.1.6 Spezialisierung von Templates ... 220
4.1.7 Partielle Spezialisierung ... 222
4.1.8 Vorgabeargumente für Templateparameter ... 224
4.1.9 Abhängige und qualifizierte Namen ... 225
4.1.10 Explizite Qualifizierung von Templates ... 230
4.1.11 Barton-Nackman-Trick ... 231
4.1.12 Das Schlüsselwort `typename` ... 233
4.1.13 Template-Templateparameter ... 235
4.1.14 Container mit Templates ... 240
4.1.15 Smart Pointer mit Templates ... 244
4.1.16 Projektorganisation mit Templates ... 250

4.2 Konzepte für den Einsatz von Templates ... 256
4.2.1 „Policy Based Design" ... 258
4.2.2 Idiome für die Template-Programmierung ... 259
4.2.3 Compiletime-Assertions ... 260
4.3 Aspektorientierte Programmierung ... 261

5 Die C++-Standardbibliothek ... 265
5.1 Die Geschichte der Standardbibliothek ... 265
5.2 Die Teilbereiche der Standardbibliothek ... 266
5.2.1 Die Streams ... 268
5.2.2 Formatierungen auf einem `ostream` ... 272
5.2.3 Die Manipulatoren ... 274
5.2.4 Die File-Streams ... 276
5.2.5 Die String-Streams ... 277
5.2.6 Die STL ... 279
5.2.7 Die Stringklasse `std::string` ... 312
5.2.8 Pseudocontainer für optimiertes Rechnen ... 317
5.2.9 Autopointer ... 327

6 Das Softwareprojekt mit C++ ... 331
6.1 Modularisierung eines C++-Projekts ... 331
6.1.1 Der Umgang mit Headerdateien ... 332
6.1.2 Namensräume ... 337
6.1.3 Das argumentenabhängige Lookup-Verfahren ... 345
6.1.4 Einige Anmerkungen zum Linker ... 348
6.1.5 Überladen der Operatoren `new` und `delete` ... 351
6.2 Persistenz und Serialisierung ... 358
6.3 Systemprogrammierung mit C++ ... 360
6.3.1 Einfaches Objektpooling ... 361
6.3.2 Nebenläufige Programmierung ... 365

Literaturverzeichnis ... 377

Listings ... 388

Sachverzeichnis ... 389