

**Richard Kaiser**  
*www.rkaiser.de*

# **C++ Programmierung**

# **Aufbaukurs**

# Inhalt

<b>1</b>	<b>Überladene Funktionen und Operatoren.....</b>	<b>7</b>
1.1	Inline-Funktionen $\Theta$ .....	7
1.2	Überladene Funktionen.....	8
1.2.1	Funktionen, die nicht überladen werden können.....	10
1.2.2	Regeln für die Auswahl einer passenden Funktion.....	10
1.3	Überladene Operatoren mit globalen Operatorfunktionen.....	14
1.3.1	Globale Operatorfunktionen.....	16
1.3.2	Die Ein- und Ausgabe von selbst definierten Datentypen.....	18
1.4	Referenztypen, Werte- und Referenzparameter.....	19
1.4.1	Werteparameter.....	19
1.4.2	Referenztypen.....	20
1.4.3	Referenzparameter.....	21
1.4.4	Referenzen als Rückgabetypen.....	22
1.4.5	Konstante Referenzparameter.....	24
<b>2</b>	<b>Namensbereiche.....</b>	<b>27</b>
2.1	Die Definition von Namensbereichen.....	28
2.2	Die Verwendung von Namen aus Namensbereichen.....	30
2.3	Header-Dateien und Namensbereiche.....	32
2.4	Aliasnamen für Namensbereiche $\Theta$ .....	34
<b>3</b>	<b>Exception-Handling.....</b>	<b>37</b>
3.1	Die <i>try</i> -Anweisung.....	38
3.2	Exception-Handler und Exceptions der Standardbibliothek.....	41
3.3	<i>throw</i> -Ausdrücke und selbst definierte Exceptions.....	43
3.4	Fehler und Exceptions.....	48
3.5	Die Freigabe von Ressourcen bei Exceptions: RAII.....	49
3.6	Exceptions in Konstruktoren und Destruktoren.....	51
3.7	<i>noexcept</i> .....	55
3.8	Die Exception-Klasse <i>system_error</i> $\Theta$ .....	56
<b>4</b>	<b>Containerklassen der C++-Standardbibliothek.....</b>	<b>59</b>
4.1	Sequenzielle Container der Standardbibliothek.....	59
4.1.1	Die Container-Klasse <i>vector</i> .....	59
4.1.2	Iteratoren.....	62
4.1.3	Geprüfte Iteratoren (Checked Iterators).....	65
4.1.4	Die bereichsbasierte <i>for</i> -Schleife.....	66
4.1.5	Iteratoren und die Algorithmen der Standardbibliothek.....	68
4.1.6	Die Speicherverwaltung bei Vektoren $\Theta$ .....	70
4.1.7	Mehrdimensionale Vektoren $\Theta$ .....	71
4.1.8	Die Container-Klassen <i>list</i> und <i>deque</i> .....	72
4.1.9	Gemeinsamkeiten und Unterschiede der sequenziellen Container.....	73
4.1.10	Die Container-Adapter <i>stack</i> , <i>queue</i> und <i>priority_queue</i> $\Theta$ .....	74

4.1.11	Container mit Zeigern.....	75
4.1.12	<code>std::array</code> - Array Container fester Größe $\Theta$ .....	75
4.2	Assoziative Container .....	76
4.2.1	Die Container <i>set</i> und <i>multiset</i> .....	77
4.2.2	Die Container <i>map</i> und <i>multimap</i> .....	77
4.2.3	Iteratoren der assoziativen Container .....	79
4.2.4	Ungeordnete Assoziative Container (Hash-Container).....	80
<b>5</b>	<b>Funktoren, Funktionsobjekte und Lambda-Ausdrücke.....</b>	<b>85</b>
5.1	Funktionen als Objekte und Parameter mit <code>std::function</code> .....	85
5.2	Der Aufrufoperator ().....	88
5.3	Prädikate und Vergleichsfunktionen .....	90
5.4	Binder $\Theta$ .....	94
5.5	Lambda-Ausdrücke.....	97
5.6	Lambda-Ausdrücke – Weitere Konzepte $\Theta$ .....	102
5.6.1	Lambda-Ausdrücke werden zu Funktionsobjekten .....	102
5.6.2	Nachstehende Rückgabetypen .....	103
5.6.3	Generische Lambda-Ausdrücke .....	104
5.6.4	Lambda-Ausdrücke höherer Ordnung $\Theta$ .....	104
5.7	Kompatible function-Typen: Kovarianz und Kontravarianz $\Theta$ .....	104
<b>6</b>	<b>Templates und die STL.....</b>	<b>107</b>
6.1	Generische Funktionen: Funktions-Templates .....	108
6.1.1	Die Deklaration von Funktions-Templates mit Typ-Parametern .....	108
6.1.2	Spezialisierungen von Funktions-Templates .....	109
6.1.3	Funktions-Templates mit Nicht-Typ-Parametern.....	114
6.1.4	Explizit instanziierte Funktions-Templates $\Theta$ .....	116
6.1.5	Explizit spezialisierte und überladene Templates .....	116
6.1.6	Rekursive Funktions-Templates $\Theta$ .....	119
6.1.7	Variadische Templates .....	120
6.2	Generische Klassen: Klassen-Templates.....	122
6.2.1	Die Deklaration von Klassen-Templates mit Typ-Parametern.....	123
6.2.2	Spezialisierungen von Klassen-Templates.....	123
6.2.3	Klassen-Templates mit Nicht-Typ-Parametern.....	128
6.2.4	Explizit instanziierte Klassen-Templates $\Theta$ .....	129
6.2.5	Partielle und vollständige Spezialisierungen $\Theta$ .....	130
6.2.6	Vererbung mit Klassen-Templates $\Theta$ .....	134
6.2.7	Tupel mit <code>&lt;tuple&gt;</code> $\Theta$ .....	135
6.2.8	Alias Templates $\Theta$ .....	136
6.3	Type Traits .....	138
6.3.1	Prüfungen bei der Kompilation: <code>static_assert</code> .....	138
6.3.2	type traits und <code>static_assert</code> .....	139
6.3.3	Eine Konstruktion von type traits.....	141
6.3.4	Die type traits Kategorien.....	142
6.3.5	type traits zur Steuerung der Übersetzung und Optimierung .....	143
6.4	Typ-Inferenz: Mit <code>decltype</code> den Datentyp eines Ausdrucks bestimmen .....	144
6.5	Kovarianz und Kontravarianz.....	146
<b>7</b>	<b>STL-Algorithmen und Lambda-Ausdrücke.....</b>	<b>149</b>
7.1	Iteratoren .....	149
7.1.1	Die verschiedenen Arten von Iteratoren .....	150
7.1.2	Umkehriteratoren .....	151
7.1.3	Einfügefunktionen und Einfügeiteratoren.....	152
7.1.4	Stream-Iteratoren .....	153
7.1.5	Container-Konstruktoren mit Iteratoren .....	154
7.2	Globale Iterator-Funktionen $\Theta$ .....	155
7.3	Lineares Suchen.....	157
7.4	Zählen .....	158

7.5	Der Vergleich von Bereichen .....	159
7.6	Suche nach Teilfolgen.....	160
7.7	Minimum und Maximum .....	160
7.8	Mit <i>all_of</i> , <i>any_of</i> , <i>none_of</i> alle Elemente in einem Bereich prüfen.....	161
7.9	Kopieren und Verschieben von Bereichen .....	161
7.10	Elemente transformieren und ersetzen .....	163
7.11	Elementen in einem Bereich Werte zuweisen $\Theta$ .....	164
7.12	Elemente entfernen – das <i>erase-remove</i> Idiom.....	165
7.13	Die Reihenfolge von Elementen vertauschen .....	167
7.13.1	Elemente vertauschen.....	167
7.13.2	Permutationen $\Theta$ .....	168
7.13.3	Die Reihenfolge umkehren und Elemente rotieren $\Theta$ .....	169
7.13.4	Elemente durcheinander mischen $\Theta$ .....	169
7.14	Algorithmen zum Sortieren und für sortierte Bereiche .....	169
7.14.1	Partitionen $\Theta$ .....	169
7.14.2	Bereiche sortieren.....	170
7.14.3	Binäres Suchen in sortierten Bereichen .....	173
7.14.4	Mischen von sortierten Bereichen .....	173
7.15	Numerische Berechnungen.....	174
7.15.1	Verallgemeinerte numerische Algorithmen .....	174
7.15.2	Valarrays $\Theta$ .....	177
7.15.3	Zufallszahlen mit <code>&lt;random&gt;</code> $\Theta$ .....	178
7.15.4	Komplexe Zahlen $\Theta$ .....	179
7.15.5	Numerische Bibliotheken neben dem C++-Standard $\Theta$ .....	182
<b>8</b>	<b>Erweiterungen der Standardbibliothek (STL).....</b>	<b>183</b>
8.1	Move-Semantik und Optimierung auf der Basis von type traits .....	183
8.2	Neue Algorithmen in C++11/14: <i>all_of</i> , <i>any_of</i> , <i>none_of</i> $\Theta$ .....	183
8.3	Erweiterungen für Strings .....	185
8.3.1	Unicode-Strings .....	185
8.3.2	Landesspezifische Einstellungen $\Theta$ .....	186
8.3.3	Reguläre Ausdrücke $\Theta$ .....	187
8.4	Erweiterungen für Container-Klassen .....	195
8.4.1	Iteratoren .....	195
8.4.2	Die Speicherverwaltung bei Vektoren $\Theta$ .....	196
8.4.3	<i>emplace</i> für Container .....	197
8.5	Erweiterungen für Stream-Klassen .....	198
8.6	Ungeordnete Assoziative Container (Hash-Container) .....	198
8.7	<i>std::array</i> - Array Container fester Größe $\Theta$ .....	201
8.8	Filesystem .....	202
<b>9</b>	<b>C++11 Smart Pointer: <i>shared_ptr</i>, <i>unique_ptr</i> und <i>weak_ptr</i>.....</b>	<b>205</b>
9.1	Gemeinsamkeiten von <i>unique_ptr</i> und <i>shared_ptr</i> .....	205
9.2	<i>unique_ptr</i> .....	210
9.3	<i>shared_ptr</i> .....	211
9.4	Deleter $\Theta$ .....	214
9.5	<i>weak_ptr</i> $\Theta$ .....	215
<b>10</b>	<b>Zeiten und Kalenderdaten mit <i>chrono</i> .....</b>	<b>219</b>
10.1	Brüche als Datentypen: Das Klassen-Template <i>ratio</i> .....	219
10.2	Ein Datentyp für Zeiteinheiten: <i>duration</i> .....	220
10.3	Datentypen für Zeitpunkte: <i>time_point</i> .....	223
10.4	Uhren: <i>system_clock</i> und <i>steady_clock</i> .....	224
<b>11</b>	<b>Multithreading .....</b>	<b>227</b>
11.1	Funktionen als Threads starten .....	228

11.1.1	Funktionen mit <i>async</i> als Threads starten .....	228
11.1.2	Funktionen mit <i>thread</i> als Threads starten.....	231
11.1.3	Lambda-Ausdrücke als Threads starten.....	233
11.1.4	Zuweisungen und <i>move</i> für Threads.....	236
11.1.5	Die Klassen <i>future</i> und <i>promise</i> .....	237
11.1.6	Exceptions in Threads und ihre Weitergabe mit <i>promise</i> .....	239
11.1.7	Der Programmablauf mit <i>async</i> .....	242
11.1.8	Informationen über Threads.....	247
11.1.9	<i>Sleep</i> -Funktionen .....	250
11.1.10	Threads im Debugger .....	251
11.2	Kritische Abschnitte.....	252
11.2.1	Atomare Datentypen.....	254
11.2.2	Kritische Bereiche mit <i>mutex</i> und <i>lock_guard</i> sperren .....	255
11.2.3	Weitere Lock-Klassen: <i>unique_lock</i> und <i>shared_lock</i> .....	260
11.2.4	Weitere Mutex-Klassen .....	262
11.2.5	Deadlocks .....	264
11.2.6	<i>call_once</i> zur Initialisierung von Daten.....	266
11.2.7	Thread-lokale Daten.....	267
11.3	Bedingungsvariablen zur Synchronisation von Threads.....	267
<b>12</b>	<b>Literaturverzeichnis .....</b>	<b>271</b>
<b>13</b>	<b>Index.....</b>	<b>273</b>