



**André Willms** hat bereits während des Studiums der Allgemeinen Informatik mit dem Schreiben von Büchern zum Thema C++ begonnen. Heute ist er Autor mehrerer erfolgreicher Bücher zu C und C++. Hauptberuflich ist er IT-Trainer mit inzwischen 17 Jahren Berufserfahrung.

André Willms

# C++: Eine kompakte Einführung



dpunkt.verlag

André Willms  
info@andrewillms.de

Lektorat: Christa Preisendanz  
Copy-Editing: Ursula Zimpfer, Herrenberg  
Herstellung: Frank Heidt  
Umschlaggestaltung: Helmut Kraus, [www.exclam.de](http://www.exclam.de)  
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek  
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;  
detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN:  
Buch 978-3-86490-229-1  
PDF 978-3-86491-651-9  
ePub 978-3-86491-652-6

1. Auflage 2015  
Copyright © 2015 dpunkt.verlag GmbH  
Wiebinger Weg 17  
69123 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

---

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Über das Buch .....	1
1.2	Vorstellung des Projekts .....	2
1.3	Identifizieren der Programmteile .....	3
1.3.1	Objekte .....	4
1.3.2	Kontrollstrukturen .....	4
1.4	Abstraktion .....	4
1.4.1	Datenabstraktion .....	5
1.4.2	Algorithmische Abstraktion .....	5
<b>2</b>	<b>Grundelemente eines C++-Programms</b>	<b>9</b>
2.1	Das erste Programm .....	9
2.1.1	Implizites return .....	10
2.2	Die Ausgabe .....	10
2.2.1	cout .....	11
2.3	include .....	13
2.4	Namensbereiche .....	14
2.5	Kommentare .....	16
2.6	Escape-Sequenzen .....	18
2.7	Zusammenfassung .....	18
2.8	Spielprojekt .....	19
<b>3</b>	<b>Arithmetik in C++</b>	<b>21</b>
3.1	Variablen .....	21
3.1.1	Integrierte Datentypen .....	22
3.1.2	Vorzeichenlose Typen .....	23
3.2	Definieren einer Variablen .....	23
3.2.1	Initialisierung .....	24
3.2.2	Lebensdauer .....	25

---

3.2.3	Automatische Typbestimmung	25
3.2.4	Definitionen zusammenfassen	26
3.3	Eingabe	27
3.4	Grundrechenarten	28
3.5	Konstanten	30
3.5.1	Konstante Ausdrücke	30
3.5.2	Unveränderliche Werte	31
3.6	Restwert	31
3.7	Verknüpfen unterschiedlicher Datentypen	32
3.8	Explizite Typumwandlung	33
3.9	Kombinierte Zuweisungsoperatoren	34
3.10	Inkrement und Dekrement	35
3.11	Mathematische Funktionen	35
3.12	Bitweise Operatoren	36
3.13	Zusammenfassung	39
3.14	Spielprojekt	40
<b>4</b>	<b>Verzweigungen</b>	<b>43</b>
4.1	Zusammengesetzte Anweisungen	43
4.2	Bedingungen	44
4.3	if	45
4.4	else	47
4.5	Logische Operatoren	48
4.6	Der ?:-Operator	51
4.7	Die Fallunterscheidung	52
4.8	static_assert	55
4.9	assert	55
4.10	Zusammenfassung	57
4.11	Spielprojekt	57
<b>5</b>	<b>Schleifen</b>	<b>59</b>
5.1	while	59
5.2	do-while	61
5.3	for	62
5.4	break	64
5.5	continue	65
5.6	Zusammenfassung	66
5.7	Spielprojekt	66

<b>6</b>	<b>Funktionen</b>	<b>67</b>
6.1	Funktionsdefinition .....	67
6.2	return .....	69
6.3	Standardwerte .....	70
6.4	Suffixrückgabetyt .....	71
6.5	Funktionsdeklaration .....	72
6.6	Module .....	72
6.7	Funktionen überladen .....	74
6.7.1	Unterscheidung in der Parameteranzahl .....	74
6.7.2	inline .....	75
6.7.3	Unterscheidung im Parametertyp .....	76
6.8	Lambda-Funktionen .....	76
6.9	Zusammenfassung .....	79
6.10	Spielprojekt .....	79
<b>7</b>	<b>Klassen</b>	<b>81</b>
7.1	Objektorientierte Programmierung .....	81
7.1.1	Objekte als abgrenzbare Einheiten .....	81
7.1.2	Nicht objektorientierte Objekte .....	82
7.2	Klassen als Bauplan .....	83
7.2.1	Definition .....	83
7.3	Zugriffsrechte .....	84
7.4	Konstruktoren .....	85
7.4.1	Standardkonstruktor .....	87
7.4.2	Der Destruktor .....	87
7.5	Methoden .....	88
7.5.1	Zugriffsmethoden .....	89
7.5.2	Konstante Methoden .....	89
7.6	Externe Definition .....	90
7.7	Mehrfachdefinition .....	92
7.8	Typalias .....	93
7.8.1	typedef .....	94
7.8.2	using .....	95
7.8.3	Zugriffsrecht .....	95
7.9	cv-Qualifizierung .....	96
7.10	Zusammenfassung .....	97
7.11	Spielprojekt .....	98

<b>8</b>	<b>Arrays und Verweise</b>	<b>101</b>
8.1	Arrays definieren . . . . .	101
8.1.1	sizeof . . . . .	102
8.2	Arbeiten mit Arrays . . . . .	102
8.2.1	Initialisierung . . . . .	103
8.2.2	Arrays durchlaufen . . . . .	103
8.3	Arrays als Funktionsparameter I . . . . .	104
8.4	Zeiger . . . . .	105
8.4.1	Hexadezimalsystem . . . . .	105
8.4.2	Der Adressoperator . . . . .	106
8.4.3	Definition eines Zeigers . . . . .	107
8.4.4	Dereferenzierung . . . . .	108
8.4.5	Zeiger als Funktionsparameter . . . . .	108
8.4.6	Zeiger auf Zeiger . . . . .	110
8.4.7	Arrays als Funktionsparameter II . . . . .	111
8.4.8	Zeigerarithmetik . . . . .	112
8.5	Referenzen . . . . .	113
8.6	Objekte als Funktionsparameter . . . . .	115
8.6.1	Referenzen auf Objekte . . . . .	115
8.6.2	Zeiger auf Objekte . . . . .	116
8.6.3	Objekte als Methodenparameter . . . . .	117
8.7	Zusammenfassung . . . . .	118
8.8	Spielprojekt . . . . .	118
<b>9</b>	<b>Strings</b>	<b>119</b>
9.1	char . . . . .	119
9.1.1	cctype . . . . .	120
9.2	C-Strings . . . . .	121
9.2.1	cstring . . . . .	123
9.2.2	Beispiel . . . . .	123
9.3	Strings . . . . .	124
9.3.1	Tastatureingabe von Strings . . . . .	125
9.3.2	Methoden von string . . . . .	126
9.4	Zusammenfassung . . . . .	128
9.5	Spielprojekt . . . . .	128
<b>10</b>	<b>Dynamische Speicherverwaltung</b>	<b>131</b>
10.1	Zeiger . . . . .	131
10.1.1	Zeiger und Konstanten . . . . .	132
10.1.2	Zeiger auf Funktionen . . . . .	133
10.1.3	Zeiger auf Klassenelemente . . . . .	134

---

10.2	Referenzen	137
10.3	new und delete	137
10.3.1	Die Klasse Name	139
10.4	Smart Pointer	140
10.4.1	Unique Pointer	140
10.4.2	Shared Pointer	143
10.4.3	Weak Pointer	145
10.4.4	Smart Pointer und Arrays	148
10.4.5	Auto-Pointer	148
10.5	Zusammenfassung	149
10.6	Spielprojekt	149
<b>11</b>	<b>Klassen – Vertiefung</b>	<b>151</b>
11.1	Reihenfolge der Zugriffsrechte	151
11.2	Der this-Zeiger	153
11.3	Konstruktoren	154
11.3.1	Standardkonstruktor	156
11.3.2	Kopierkonstruktor	157
11.3.3	Die Klasse Name	158
11.3.4	Elementinitialisierungsliste	160
11.3.5	Verschiebekonstruktor	163
11.3.6	Implizite Typumwandlung	166
11.3.7	Konstruktordelegation	166
11.4	Destruktoren	167
11.5	Konstante Objekte und Elemente	168
11.5.1	Implizite Objektparameter	168
11.5.2	mutable	170
11.6	Funktionen als deleted oder default definieren	176
11.7	Zusammenfassung	177
11.8	Spielprojekt	177
<b>12</b>	<b>Klassen – Abschluss</b>	<b>181</b>
12.1	Standardwerte für Attribute	181
12.2	Verschachtelte Klassendefinitionen	181
12.3	Statische Klassenelemente	183
12.3.1	Statische Methoden	183
12.3.2	Statische Attribute	186
12.3.3	Statische Variablen	191
12.4	Konstruktoren und ihre Anwendung	192
12.4.1	Funktionsaufruf aus Konstruktoren heraus	192
12.4.2	Unvollendet konstruierte Objekte	194



12.5	Implizite Klasselemente .....	196
12.5.1	Impliziter Standardkonstruktor .....	198
12.5.2	Impliziter Kopierkonstruktor .....	198
12.5.3	Impliziter Verschiebekonstruktor .....	199
12.5.4	Impliziter Kopier-Zuweisungsoperator .....	199
12.5.5	Impliziter Verschiebe-Zuweisungsoperator .....	200
12.5.6	Impliziter Destruktor .....	201
12.6	Aufzählungen .....	201
12.7	Zusammenfassung .....	202
12.8	Spielprojekt .....	203
<b>13</b>	<b>Namensbereiche</b>	<b>205</b>
13.1	Deklarative Bereiche, potenzielle und tatsächliche Bezugsrahmen ..	205
13.2	Namensbereiche definieren .....	208
13.3	Die Using-Direktive .....	209
13.4	Ein Alias für Namensbereiche .....	210
13.5	Unbenannte Namensbereiche .....	211
13.6	Die Using-Deklaration .....	211
13.7	Zusammenfassung .....	212
13.8	Spielprojekt .....	213
<b>14</b>	<b>Operatoren überladen</b>	<b>215</b>
14.1	Zuweisungsoperatoren .....	215
14.1.1	Kopier-Zuweisungsoperator .....	215
14.1.2	Verschiebe-Zuweisungsoperator .....	217
14.1.3	Kombinierte Zuweisungsoperatoren .....	218
14.2	Rechenoperatoren .....	219
14.2.1	Operation als Methode .....	219
14.2.2	Operation als Funktion .....	219
14.2.3	Methode oder Funktion .....	222
14.2.4	Operatoren mit Verschiebe-Semantik .....	223
14.2.5	Standardverhalten nachbilden .....	226
14.3	Vergleichsoperatoren .....	228
14.3.1	Operator-Templates .....	229
14.4	Die Operatoren << und >> .....	229
14.4.1	operator<< .....	230
14.4.2	operator>> .....	230
14.5	Der Operator [] .....	231
14.6	Der Operator () .....	232
14.7	Die Operatoren -> und * .....	233
14.8	Umwandlungsoperatoren .....	235

14.9	Die Operatoren ++ und --	236
14.9.1	Präfixoperatoren	238
14.9.2	Postfixoperatoren	238
14.9.3	Weitere Operatoren	239
14.10	Probleme mit Operatoren	240
14.11	Zusammenfassung	241
14.12	Spielprojekt	241
<b>15</b>	<b>Templates</b>	<b>243</b>
15.1	Klassen-Templates	243
15.2	Funktions-Templates	245
15.3	Template-Parameter	246
15.3.1	Standardargumente	247
15.4	Template-Spezialisierung	247
15.5	typename	248
15.6	Zusammenfassung	249
<b>16</b>	<b>STL</b>	<b>251</b>
16.1	Die Komponenten der STL	251
16.2	Container	252
16.2.1	Laufzeitklassen	252
16.2.2	Die Container im Überblick	253
16.2.3	STL-konforme Container	254
16.3	Iteratoren	255
16.3.1	Entwurf eines Iterators	255
16.3.2	Ein Iterator für Name	258
16.3.3	Iteratorkategorien	260
16.3.4	STL-konforme Iteratoren erstellen	263
16.3.5	Iteratoren erzeugen	263
16.3.6	Insert-Iteratoren	264
16.3.7	Stream-Iteratoren	267
16.4	Algorithmen	268
16.4.1	Die Algorithmen im Überblick	268
16.5	Die STL im Einsatz	271
16.5.1	Variable Funktionsaufrufe	271
16.5.2	Aufrufübergreifende Zustände	275
16.5.3	Element suchen	276
16.5.4	Element suchen mit eigener Bedingung	277
16.5.5	Elemente löschen	278
16.5.6	Elemente kopieren	278

16.5.7	Elemente sortieren .....	279
16.5.8	Eigene Listeninitialisierungsconstructoren .....	281
16.6	Zusammenfassung .....	282
16.7	Spielprojekt .....	282
<b>17</b>	<b>Vererbung I</b>	<b>287</b>
17.1	Das Klassendiagramm der UML .....	287
17.2	Vererbung in C++ .....	290
17.3	Die Vererbungssyntax .....	292
17.4	Geschützte Elemente .....	296
17.4.1	Zugriff auf Basisklassenelemente .....	298
17.5	Polymorphie .....	298
17.6	Verdecken von Methoden .....	300
17.7	Überschreiben von Methoden .....	302
17.8	Virtuelle Methoden .....	304
17.8.1	Virtuelle Methoden und Constructoren .....	305
17.8.2	Downcasts .....	307
17.8.3	Virtuelle Destruktoren .....	308
17.9	Rein-virtuelle Methoden .....	311
17.9.1	Rein-virtuelle Methoden mit Implementierung .....	311
17.9.2	Rein-virtuelle Destruktoren .....	312
17.10	Vererbung und Arrays .....	313
17.11	Vererbung und Standardwerte .....	314
17.12	Vererbung und überladene Operatoren .....	315
17.13	Versiegelte Elemente .....	316
17.13.1	Versiegelte Klasse .....	316
17.13.2	Versiegelte Methode .....	316
17.13.3	Warum Elemente versiegeln? .....	317
17.14	Geerbte Constructoren verwenden .....	318
17.15	Überschreibungshilfe .....	319
17.16	Zusammenfassung .....	320
17.17	Spielprojekt .....	320
<b>18</b>	<b>Vererbung II</b>	<b>325</b>
18.1	Beziehungen .....	325
18.1.1	ist ein .....	325
18.1.2	ist implementiert mit .....	330
18.1.3	hat ein .....	332
18.2	Was wird vererbt? .....	333
18.2.1	Schnittstelle mit verbindlicher Implementierung .....	333
18.2.2	Schnittstelle mit überschreibbarer Implementierung .....	335

18.2.3	Schnittstelle	336
18.2.4	Implementierung	338
18.3	Das Offen-geschlossen-Prinzip	338
18.4	Operationen oben, Daten unten	342
18.5	Das Umkehrung-der-Abhängigkeit-Prinzip	344
18.6	Das Einzelne-Verantwortung-Prinzip	346
18.7	Zusammenfassung	347
18.8	Spielprojekt	348
<b>19</b>	<b>Mehrfachvererbung</b>	<b>353</b>
19.1	Gemeinsame Basisklassen	354
19.2	Virtuelle Basisklassen	356
19.3	Konstruktoren virtueller Basisklassen	358
19.4	Einsatz von Mehrfachvererbung	359
19.5	Zusammenfassung	360
19.6	Spielprojekt	361
<b>20</b>	<b>Ausnahmen</b>	<b>365</b>
20.1	Warum Ausnahmen?	365
20.2	Einsatz von Ausnahmen	367
20.3	Vordefinierte Ausnahmen	370
20.3.1	Header-Datei »exception«	371
20.3.2	Header-Datei »typeinfo«	371
20.3.3	Header-Datei »memory«	371
20.3.4	Header-Datei »new«	371
20.3.5	Header-Datei »stdexcept«	372
20.4	Ausnahmen im Detail	372
20.4.1	terminate	372
20.4.2	Das Verlassen eines Try-Blocks	373
20.4.3	uncaught_exception	374
20.4.4	Das Werfen einer Ausnahme	374
20.4.5	Das Fangen einer Ausnahme	375
20.5	Ausnahmespezifikationen	378
20.5.1	Ausnahmespezifikationen in der Praxis	379
20.6	Ausnahmen und Konstruktoren	379
20.7	Ausnahmen und Destruktoren	381
20.8	Ausnahmen und dynamische Speicherverwaltung	382
20.9	Ressourcenerwerb ist Initialisierung	384
20.10	Funktions-Try-Blöcke	385
20.11	Ausnahmensicherheit	387
20.12	Zusammenfassung	387

---

<b>21</b>	<b>Das Spiel</b>	<b>389</b>
21.1	Aktionen .....	389
21.2	Spielregeln .....	392
21.3	Räume .....	394
21.4	Die Erzeugung der Spielwelt .....	396
21.4.1	Erstellen der Räume .....	396
21.4.2	Erzeugen der Gegenstände .....	397
21.4.3	Erstellen der Ausgänge .....	398
21.4.4	Erzeugen von Türen .....	399
21.4.5	Erstellen der Interaktionen .....	402
	<b>Literatur</b>	<b>405</b>
	<b>Index</b>	<b>407</b>