

Tim Scheiter

Eintauchen in C++



begleitende Inhalte auf
www.cplusplus-buch.de/eintauchen

BoD – Books on Demand, Norderstedt

Deutsche Fassung mit deutschen Quellcode-Kommentaren

Inhalt

1 Über dieses Buch	19
1.1 Was gelehrt wird und was nicht	19
1.2 Vorkenntnisse und Voraussetzungen	20
1.3 Formatierungen	21
1.3.1 Schlüsselwörter, Namen und Beispiele	21
1.3.2 Syntax der Quellcodes	21
1.3.3 Verweise auf Kapitel	22
1.4 Code-Beispiele in diesem Buch	22
1.4.1 Code-Ausschnitte (Listings)	22
1.4.2 Programmierung eigener Klassen	23
1.4.3 Downloads	24
1.5 Verwendete Abbildungen und Tabellen	24
1.6 Zahlenwerte	24
2 Begriffsklärungen	25
2.1 Über dieses Kapitel	25
2.2 Entwicklung von Code	25
2.2.1 Quellcodes und Quelldateien	25
2.2.2 Syntax und Semantik	26
2.2.3 Compiler und Linker	26
2.2.4 Definition und Deklaration	28
2.2.5 Implementierung und Initialisierung	29
2.3 Objektorientierung	30
2.3.1 Klassen, Objekte und Instanzen	30
2.3.2 Kapselung von Daten	31
2.3.3 Primitive und komplexe Datentypen	32
2.4 Speicherverwaltung	33
2.4.1 Stack und Heap	33
2.4.2 Freigabe von Speicher	33
2.5 Die imperative Programmierung	34

3 Erweiterte Grundlagen	35
3.1 Starten mit der main-Methode	35
3.2 Eingebaute Datentypen in C++	36
3.2.1 Allgemeines zu den Datentypen	36
3.2.2 Logischer Datentyp (bool)	36
3.2.3 Ganzzahlige Typen	37
3.2.4 Die Schlüsselwörter signed und unsigned	38
3.2.5 Gleitkommatypen	39
3.2.6 Druckbare Zeichen (char)	40
3.3 Deklaration und Initialisierung von Variablen	41
3.4 Verwendung von Konstanten	45
3.5 Arithmetik und Operatoren	46
3.5.1 Allgemeines zu den Operatoren	46
3.5.2 Binäre Operatoren	47
3.5.3 Unäre Vorzeichen	50
3.5.4 Inkrement und Dekrement	51
3.5.5 Binäre Zuweisungsoperatoren	53
3.5.6 Klammerung von Ausdrücken	54
3.5.7 Sequenzen	56
3.6 Arrays fester Größen	57
3.7 Whitespaces	62
3.8 Typdefinitionen	63
3.9 Die Schlüsselwörter register und volatile	64
4 Präprozessor	65
4.1 Der Nutzen des Präprozessors	65
4.2 Verwendung von Direktiven	66
4.2.1 Allgemeines zu den Präprozessor-Direktiven	66
4.2.2 Inkludieren von Header-Dateien (#include)	67
4.2.3 Definition von Symbolen (#define und #undef)	70
4.2.4 Bedingter Code (#ifdef , #ifndef und #endif)	73
4.2.5 Verzweigter Programmcode (#else)	76
4.2.6 Die defined -Direktive (#if und #elif)	77
4.2.7 Vordefinierte Symbole	80
4.2.8 Weitere Direktiven	81
4.3 Header-Guards	83

5 Klassen & Bezugsrahmen (Scoping)	85
5.1 Die Schlüsselwörter class und struct	85
5.2 Deklaration von Klassennamen	85
5.3 Definition von Klassen	86
5.3.1 Bezugsrahmen von Klassendefinitionen	86
5.3.2 Eigenschaften in Klassen	88
5.3.3 Sichtbarkeitsbereiche in Klassen	89
5.3.4 Der Unterschied von class zu struct	92
5.4 Bildung von Instanzen	92
5.5 Zugriffe über den Punktoperator	95
5.6 Eingebettete Objekte	96
5.7 Scoping durch Bezugsrahmen	98
5.7.1 Grundlagen des Scopings	98
5.7.2 Verschachtelte Bereiche	99
5.7.3 Der Scope-Operator	100
5.7.4 Scoping in globalen Bereichen	100
5.7.5 Globale Methoden	103
5.7.6 Lokales Scoping	107
5.7.7 Klassenbereiche	109
5.7.8 Übersicht aller Scopes	112
5.7.9 Zugriffe auf globale Daten	113
5.8 Lokale und globale Gültigkeit	114
5.9 Der Aufzählungstyp enum	114
5.10 Lebensdauer von Objekten	120
5.10.1 Lebensdauer in Scopes	120
5.10.2 Das Schlüsselwort static	123
5.10.3 Statische Eigenschaften in Klassen	126
5.10.4 Konstante und statische Eigenschaften	129
5.11 Kontrollstrukturen	130
5.11.1 Grundlagen von Programmabläufen	130
5.11.2 Binäre Vergleichsoperatoren	130
5.11.3 Bedingte Anweisung (if und else)	133
5.11.4 Zählschleife (for)	137
5.11.5 Kopfgesteuerte Schleife (while)	140
5.11.6 Fußgesteuerte Schleife (do-while)	141
5.11.7 Abbrüche von Schleifendurchläufen	142
5.11.8 Fallunterscheidung (switch-case-default)	143

5.12	Typdefinitionen mit Klassen	148
5.13	Der ternäre Operator	149
5.14	Arrays fester Größen und Objekte	151
5.15	Überblick der Operatoren des Kapitels	152
6 Namensräume		153
6.1	Was Namensräume darstellen	153
6.2	Der globale Namensraum	154
6.3	Definition von Namensräumen	154
6.3.1	Das Schlüsselwort namespace	154
6.3.2	Verschachtelung von Namensräumen	156
6.4	Identifikation von Daten im Namensraum	160
6.4.1	Explizite Zugriffe über den Scope-Operator	160
6.4.2	Die using -Direktive	161
6.4.3	Namensräume ohne Identifikatoren	163
6.5	Aliasing von Namensräumen	165
6.6	Mehrdeutigkeit und Namenskollisionen	168
7 Bitmanipulation & Logik		171
7.1	Bit, Byte und Datenspeicherung	171
7.2	Interne Darstellung von Daten	173
7.2.1	Allgemeines zur Codierung	173
7.2.2	Speichergrößen ganzzahliger Typen	174
7.2.3	Darstellung vorzeichenloser Datentypen	175
7.2.4	Codierung vorzeichenbehafteter Datentypen	177
7.2.5	Mantisse und Exponent (Gleitkommatypen)	180
7.2.6	NaN und Inf (Gleitkommatypen)	183
7.2.7	Signifikante Bit (MSB und LSB)	185
7.3	Daten zusammenhängender Speicherbereiche	186
7.3.1	Abbildung von Objekten und Arrays	186
7.3.2	Speicherausrichtung und Füllbyte	188
7.4	Der sizeof -Operator	192

7.5	Komprimierte Datenspeicherung	197
7.5.1	Der Datenverbund union	197
7.5.2	Anonyme Datenverbunde	201
7.5.3	Bitfelder in Eigenschaften	203
7.6	Anonyme Klassen und Enumerationen	208
7.7	Bitweise Operationen	208
7.7.1	Übersicht der Operatoren	208
7.7.2	Bitverschiebungen	209
7.7.3	Bitweise Verknüpfungen	213
7.7.4	Das Einerkomplement	218
7.7.5	Verwendung von Wahrheitswerten	219
7.7.6	Bitweise Zuweisungsoperatoren	220
7.8	Über- und Unterläufe ganzzahliger Bereiche	222
7.9	Promotion und Demotion primitiver Typen	224
7.10	Implizite Typkonvertierungen	225
7.10.1	Konvertierung primitiver Typen	225
7.10.2	Konvertierung nach bool	228
7.11	Explizite Typkonvertierungen	230
7.11.1	Literale und Suffixe	230
7.11.2	Konstruktoren primitiver Typen	233
7.11.3	Der Konvertierungsoperator	234
7.12	Aussagenlogik	235
7.12.1	Logische Verknüpfungen	235
7.12.2	Der Negierungsoperator	238
7.12.3	Überblick der Operatoren	239
7.13	Verknüpfte Bedingungen im Präprozessor	240
8	Methoden in Klassen	241
8.1	Inhalte dieses Kapitels	241
8.2	Deklaration von Methoden	241
8.2.1	Signaturen von Methoden	241
8.2.2	Der const -Qualifizierer	242
8.2.3	Prototypen von Methoden	243

8.3	Implementierung von Methoden	244
8.3.1	Ein bereits Bekannter: Der Scope-Operator	244
8.3.2	Methoden in Namensräumen	246
8.3.3	inline -Methoden	248
8.3.4	Rückgabe von Werten	249
8.4	Parameter von Methoden	253
8.4.1	Parameterlisten	253
8.4.2	Die lokale Kopie (call-by-value)	254
8.4.3	Der Standardparameter	256
8.5	Getter- und Setter-Methoden (Point2D-Praxis)	258
8.6	Logische Methoden	260
8.7	Konstruktoren	261
8.7.1	Verwendung von Konstruktoren	261
8.7.2	Initialisierungslisten	262
8.7.3	Der Standardkonstruktor	264
8.7.4	Das Schlüsselwort explicit	267
8.7.5	Konstruktoraufrufe eingebetteter Objekte	271
8.8	Destruktoren	273
8.8.1	Verwendung von Destruktoren	273
8.8.2	Explizite Destruktoraufrufe	274
8.9	Statische Methoden	275
8.10	Das Überladen von Methoden	279
8.11	Mehrdeutige Methodenaufrufe	280
8.11.1	Mehrdeutigkeit durch Methodenüberladung	280
8.11.2	Mehrdeutigkeit durch Standardparameter	283
8.11.3	Mehrdeutigkeit durch Überladung und Parameter	284
8.12	Rekursion von Methoden	286
8.13	Freundschaft von Methoden	289
8.13.1	friend -Methoden und friend -Klassen	289
8.13.2	Globale friend -Methoden	292
8.14	Delegation an Methoden und Seiteneffekte	293
8.15	Objekte, Methoden und das Schlüsselwort const	295
8.16	Kapselung von Konstruktoren (Wrapper-Klassen)	297
8.17	Methoden im Datenverbund (union)	298

9 Die Praxisklasse Number	301
9.1 Was Sie in diesem Kapitel erwartet	301
9.2 Vorarbeit für die Klassenprogrammierung	301
9.2.1 Primitive Datentypen	301
9.2.2 Ein Aufzählungstyp	303
9.2.3 Definition des Datenverbundes	305
9.3 Definition der Klasse	306
9.4 Prototypen der Klasse	307
9.4.1 Prototypen überladener Konstruktoren	307
9.4.2 Prototypen der Getter- und Setter-Methoden	308
9.4.3 Prototypen logischer Methoden	309
9.4.4 Prototyp einer statischen Methode	310
9.5 Implementierungen der Klasse	310
9.5.1 Implementierung der Konstruktoren	310
9.5.2 Implementierung der Getter-Methoden	313
9.5.3 Implementierung der Setter-Methoden	315
9.5.4 Implementierung der logischen Methoden	317
9.5.5 Implementierung der statischen Methode	319
9.6 Statische und konstante Eigenschaften	321
9.7 Speicherbelegung	323
9.8 Verwendung der Klasse	325
10 Zeiger & Referenzen	327
10.1 Die Bedeutung der Zeiger	327
10.2 Deklaration von Zeigervariablen	328
10.2.1 Deklaration eindimensionaler Zeiger	328
10.2.2 Deklaration mehrdimensionaler Zeiger	329
10.2.3 Deklaration typloser Zeiger	330
10.2.4 Zeiger in Deklarationslisten	330
10.3 Der Adressoperator	331
10.4 Zuweisung und Initialisierung von Zeigern	332
10.5 Nullzeiger und Zeigervalidierung	333
10.6 Zeiger mit dem Schlüsselwort const	336

10.7 Dereferenzierung von Zeigern	339
10.8 Zeiger und Objekte	341
10.8.1 Deklaration und Initialisierung	341
10.8.2 Zugriffe über den Pfeiloperator	342
10.8.3 Zeiger auf konstante Objekte	343
10.8.4 Konstante Zeiger auf konstante Objekte	344
10.8.5 Der this -Zeiger	345
10.9 Typdefinitionen mit Zeigern	346
10.10 Der sizeof -Operator mit Zeigern	347
10.11 Zeiger und Methoden	348
10.11.1 Zeiger als Rückgabewerte	348
10.11.2 Zeiger als Methodenparameter	349
10.11.3 Zeiger als Standardparameter	352
10.12 Dynamische Speicherallokierung	352
10.12.1 Die Operatoren new und delete	352
10.12.2 Gefährliche Verwendung des delete -Operators	358
10.12.3 Zeiger auf dem Heap	359
10.12.4 Der new -Operator in Methoden	361
10.13 Vergleiche von Zeigervariablen	363
10.14 Vagabundierende Zeiger	364
10.15 Zeiger und Arrays	367
10.15.1 Arrays fester Größen und Zeiger	367
10.15.2 Arrays fester Größen als Parameter	370
10.15.3 Dynamische Arrays (new[] und delete[])	375
10.15.4 Zugriffe über Indizierung	379
10.15.5 Zeigerarithmetik	382
10.15.6 Speicherlecks durch den delete[] -Operator	388
10.15.7 Mehrdimensionale und dynamische Arrays	389
10.16 Zeiger als Eigenschaften	394
10.16.1 Deklaration in Klassen (Line2D-Praxis)	394
10.16.2 Bildung und Freigabe innerer Instanzen	395
10.16.3 Verwendung von Objekten über Zeiger-Member	399
10.16.4 Speicherbedarf innerer Instanzen	401
10.16.5 Flache und tiefe Kopien	403
10.17 Übersicht der Zeiger-Operatoren	405
10.18 Das Schlüsselwort nullptr	406

10.19	Die Bedeutung der Referenzen	407
10.20	Deklaration und Verwendung von Referenzen	408
10.20.1	Initialisierung von Referenzen	408
10.20.2	Zugriffe auf Variablen über Referenzen	409
10.20.3	Referenzen auf Referenzen	411
10.21	Typdefinitionen mit Referenzen	412
10.22	Referenzen mit dem const -Qualifizierer	412
10.23	Referenzen und Methoden	414
10.23.1	Referenzen als Parameter (call-by-reference)	414
10.23.2	Konstante Referenzen in Parametern	417
10.23.3	Objekt-Referenzen als Rückgabewerte	420
10.24	Flache und tiefe Kopien von Objekten	421
10.24.1	Allgemeines zu Objektkopien	421
10.24.2	Der Standard-Kopierkonstruktor	422
10.24.3	Eigene Kopierkonstruktoren der Praxisklassen	424
10.24.4	Der Standard-Zuweisungsoperator	429
10.24.5	Überschriebene Zuweisungsoperatoren	431
10.25	Vagabundierende Referenzen	436
11 Zeichenketten		439
11.1	Was Ihnen dieses Kapitel lehrt	439
11.2	Deklaration und Verwendung von Zeichenketten	440
11.2.1	Druckbare Zeichen in Arrays fester Größen	440
11.2.2	Zeichenketten-Literale	441
11.2.3	Dynamische Zeichenketten	442
11.2.4	Zeichenketten in zwei Dimensionen	443
11.3	Null-Terminierung von Zeichenketten	445
11.4	Interne Darstellung der Zeichenketten-Literale	448
11.5	Vergleiche von Zeichen und Zeichenketten	452
11.6	Erweiterung der Number-Praxisklasse	455
11.6.1	Prototyp einer neuen Methode	455
11.6.2	Implementierung der neuen Methode	455
11.6.3	Verwendung der Methode	458

11.7 Zeichenketten-Literale im Präprozessor	458
11.8 Parameter der main-Methode	460
12 Vererbung von Klassen	463
12.1 Grundlagen der Vererbung	463
12.2 Definition abgeleiteter Klassen	464
12.2.1 Kindklassen durch Vererbung	464
12.2.2 Vererbung von Sichtbarkeit	468
12.2.3 Das Schlüsselwort protected	470
12.2.4 Der Unterschied von class zu struct	471
12.2.5 Die Point3D-Praxisklasse	472
12.3 Vererbungshierarchien	473
12.4 Speicherbedarf von Instanzen geerbter Klassen	475
12.5 Konstruktoren in abgeleiteten Klassen	476
12.5.1 Aufrufe von Basiskonstruktoren	476
12.5.2 Konstruktoraufrufe im Programmablauf	480
12.5.3 Abstrakte Basisklassen	481
12.6 Destruktoren geerbter Klassen	482
12.7 Methoden in abgeleiteten Klassen	483
12.7.1 Erweiterte Funktionalität der Kindklassen	483
12.7.2 Das Überschreiben von Methoden	485
12.7.3 Überschreiben vs. Überladen	489
12.8 Statische Elemente in geerbten Klassen	490
12.9 Vererbung und das Schlüsselwort friend	492
12.10 Polymorphie	493
12.10.1 Zeiger auf Instanzen abgeleiteter Klassen	493
12.10.2 Virtuelle Methoden und Destruktoren (virtual)	494
12.10.3 Abstraktion durch reinvirtuelle Methoden	498
12.10.4 Polymorphe Zeiger als Parameter	500
12.10.5 Polymorphe Referenzen	501
12.11 Finalisierung	502