

PROGRAMMER'S CHOICE



Scott Meyers

Effektiv C++ programmieren

55 Möglichkeiten, Ihre Programme
und Entwürfe zu verbessern



3. Auflage

Studentenausgabe



ADDISON-WESLEY



Scott Meyers

Effektiv C++ programmieren

Dritte Ausgabe

**55 Möglichkeiten, Ihre Programme und
Entwürfe zu verbessern**



ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Inhalt

Vorwort	11
Danksagung	13
Einleitung	17
Terminologie	19
Namenskonventionen	24
Threads	25
TR1 und Boost	25
I Machen Sie sich mit C++ vertraut	27
1.1 Tipp 1: Betrachten Sie C++ als eine Kombination von Sprachen	27
1.2 Tipp 2: Ziehen Sie const, enum und inline gegenüber #define vor	29
1.3 Tipp 3: Verwenden Sie nach Möglichkeit const	33
1.3.1 Konstante Elementfunktionen	36
1.3.2 Die Duplizierung in konstanten und nicht konstanten Elementfunktionen vermeiden	39
1.4 Tipp 4: Initialisieren Sie Objekte vor ihrer Verwendung	43
2 Konstruktoren, Destruktoren und Zuweisungsoperatoren	51
2.1 Tipp 5: Lernen Sie die Funktion kennen, die C++ implizit schreibt und aufruft	51
2.2 Tipp 6: Untersagen Sie ausdrücklich die Verwendung unerwünschter compilergenerierter Funktionen	54
2.3 Tipp 7: Deklarieren Sie Destruktoren virtuell in polymorphen Basisklassen	57
2.4 Tipp 8: Verhindern Sie, dass Ausnahmen Destruktoren überschreiten	62
2.5 Tipp 9: Rufen Sie während der Konstruktion oder Zerstörung niemals virtuelle Funktionen auf	66
2.6 Tipp 10: Lassen Sie Zuweisungsoperatoren einen Verweis auf *this zurückgeben	71
2.7 Tipp 11: Führen Sie die Selbstzuweisung in operator= durch	72
2.8 Tipp 12: Kopieren Sie alle Teile eines Objekts	76

3	Ressourcenverwaltung	81
3.1	Tipp 13: Verwenden Sie Objekte zur Ressourcenverwaltung	81
3.2	Tipp 14: Bedenken Sie das Kopierverhalten in Klassen zur Ressourcenverwaltung	87
3.3	Tipp 15: Bieten Sie in Klassen zur Ressourcenverwaltung einen Zugriff auf Rohressourcen an	90
3.4	Tipp 16: Verwenden Sie für einander entsprechende Anwendungen von new und delete die gleiche Form	94
3.5	Tipp 17: Speichern Sie mit new erzeugte Objekte in intelligenten Zeigern eigenständiger Anweisungen	97
4	Design und Deklaration	101
4.1	Tipp 18: Erleichtern Sie die korrekte und erschweren Sie die falsche Verwendung von Schnittstellen	101
4.2	Tipp 19: Behandeln Sie Klassendesign als Typdesign	107
4.3	Tipp 20: Übergeben Sie Objekte lieber als Verweis auf const statt als Wert	110
4.4	Tipp 21: Geben Sie keine Verweise zurück, wenn Sie ein Objekt zurückgeben müssen	114
4.5	Tipp 22: Deklarieren Sie Datenelemente als privat	119
4.6	Tipp 23: Ziehen Sie nicht befreundete Nichtelement-funktionen den Elementfunktionen vor	122
4.7	Tipp 24: Deklarieren Sie Nichtelementfunktionen bei Typumwandlungen für alle Parameter	127
4.8	Tipp 25: Unterstützen Sie die ausnahmsfreie swap-Funktion	131
5	Implementierung	139
5.1	Tipp 26: Schieben Sie die Variablendefinition so lange wie möglich auf	139
5.2	Tipp 27: Minimieren Sie die explizite Typumwandlung	142
5.3	Tipp 28: Geben Sie keine »Handles« für interne Daten eines Objekts zurück	150
5.4	Tipp 29: Streben Sie ausnahmesicheren Code an	153
5.5	Tipp 30: Lernen Sie die Einzelheiten von Inline-Funktionen kennen	161
5.6	Tipp 31: Minimieren Sie die Kompilierungsabhängigkeiten zwischen Dateien	167
6	Vererbung und objektorientiertes Design	177
6.1	Tipp 32: Modellieren Sie bei der öffentlichen Vererbung die Beziehung »ist ein«	178
6.2	Tipp 33: Verdecken Sie keine geerbten Namen	184
6.3	Tipp 34: Unterscheiden Sie zwischen der Vererbung der Schnittstelle und der Implementierung	190
6.4	Tipp 35: Verwenden Sie Alternativen zu virtuellen Funktionen	198
6.5	Tipp 36: Definieren Sie geerbte nicht virtuelle Funktionen nicht neu	207
6.6	Tipp 37: Definieren Sie den geerbten Standard-parameterwert einer Funktion nicht neu	210
6.7	Tipp 38: Modellieren Sie die Beziehungen »hat ein« und »ist implementiert in Form von« durch Komposition	214

6.8	Tipp 39: Seien Sie bei der privaten Vererbung vorsichtig	217
6.9	Tipp 40: Seien Sie bei der mehrfachen Vererbung vorsichtig	223
7	Templates und generische Programmierung	231
7.1	Tipp 41: Machen Sie sich mit impliziten Schnittstellen und Polymorphismus zur Kompilierungszeit vertraut	231
7.2	Tipp 42: Lernen Sie die zwei Bedeutungen von typename kennen	235
7.3	Tipp 43: Lernen Sie den Zugriff auf Namen in Template-Basisklassen	240
7.4	Tipp 44: Entfernen Sie parameterunabhängigen Code aus Templates	245
7.5	Tipp 45: Verwenden Sie Elementfunktions-Templates für »alle kompatiblen Typen«	251
7.6	Tipp 46: Definieren Sie bei erforderlichen Typumwandlungen Nichtelementfunktionen innerhalb von Templates	256
7.7	Tipp 47: Verwenden Sie Traits-Klassen für Informationen über Typen	260
7.8	Tipp 48: Beachten Sie die Template-Metaprogrammierung	267
8	Anpassung von new und delete	275
8.1	Tipp 49: Lernen Sie das Verhalten des new-Handlers kennen	276
8.2	Tipp 50: Erkennen Sie, wann es sinnvoll ist, new und delete zu ersetzen	284
8.3	Tipp 51: Halten Sie sich beim Schreiben von new und delete an die üblichen Vereinbarungen	289
8.4	Tipp 52: Verwenden Sie bei Placement-new auch Placement-delete	293
9	Verschiedenes	301
9.1	Tipp 53: Beachten Sie Compilerwarnungen	301
9.2	Tipp 54: Machen Sie sich mit der Standardbibliothek einschließlich TR1 vertraut	302
9.3	Tipp 55: Machen Sie sich mit Boost vertraut	308
A	Weiterführende Lektüre	313
B	Zuordnung der Tipps zur vorhergehenden Ausgabe	315
	Index	317