

THINK LIKE A PROGRAMMER

Typische Programmieraufgaben
kreativ lösen am Beispiel von C++



Inhaltsverzeichnis

	Danksagungen	11
	Einführung	13
	Über dieses Buch	15
	Voraussetzungen	16
	Ausgewählte Themen	16
	Programmierstil	16
	Übungen	17
	Warum C++?	17
I	Strategien zur Problemlösung	19
I.1	Klassische Rätsel	20
I.1.1	Fuchs, Gans und Getreidesack	21
	Aufgabe: Wie kann der Fluss überquert werden?	21
I.1.2	Schiebepuzzles	25
	Aufgabe: Verschieben der 8	26
	Aufgabe: Verschieben der 5	27
I.1.3	Sudoku	30
	Aufgabe: Vervollständigen eines Sudoku-Quadrats	31
I.1.4	Das Quarrasi-Schloss	33
	Aufgabe: Öffnen des außerirdischen Schlosses	33
I.2	Gängige Verfahren zur Problemlösung	36
I.2.1	Niemals ohne Plan	37
I.2.2	Umformulieren der Aufgabenstellung	38
I.2.3	Zerlegung in Teilaufgaben	39
I.2.4	Mit Bekanntem anfangen	40
I.2.5	Reduktion der Problemstellung	41
I.2.6	Analogien suchen	42
I.2.7	Experimentieren	43
I.2.8	Nicht entmutigen lassen	44
I.3	Übungen	45
2	Wahre Rätsel	47
2.1	Verwendete C++-Syntax	47

2.2	Ausgabe von Mustern	48
	Aufgabe: Halbiertes Quadrat	48
	Aufgabe: Quadrat (Halbiertes Quadrat, Reduktion)	48
	Aufgabe: Zeile (Halbiertes Quadrat, weitere Reduktion)	49
	Aufgabe: Herunterzählen durch Hochzählen	50
	Aufgabe: Hochkant stehendes Dreieck	51
2.3	Eingabeverarbeitung	54
	Aufgabe: Test einer Luhn-Prüfsumme	54
2.3.1	Zerlegung der Aufgabenstellung	56
	Aufgabe: Ziffer in Ganzzahl konvertieren	58
	Aufgabe: Test einer Luhn-Prüfsumme fester Länge	60
	Aufgabe: Test einer einfachen Prüfsumme fester Länge	60
	Aufgabe: Positiv oder negativ	63
2.3.2	Zusammenstellen der Teillösungen	64
2.4	Statusverfolgung	66
	Aufgabe: Entschlüsseln einer Botschaft	66
	Aufgabe: Einlesen einer Zahl mit drei oder vier Ziffern	71
	Aufgabe: Einlesen einer Zahl mit drei oder vier Ziffern, weiter vereinfacht	72
2.5	Fazit	81
2.6	Übungen	81
3	Arrays	85
3.1	Array-Grundlagen	86
3.1.1	Speichern	86
3.1.2	Kopieren	87
3.1.3	Zugriff und Suche	88
3.1.4	Sortieren	89
3.1.5	Statistische Werte	92
3.2	Aufgabenstellungen mit Arrays	93
	Aufgabe: Modalwert berechnen	93
3.2.1	Refactoring	97
3.3	Arrays mit fest vorgegebenen Daten	100
3.4	Nicht-skalare Arrays	102
3.5	Mehrdimensionale Arrays	104
3.6	Wann werden Arrays verwendet?	108
3.7	Übungen	113
4	Zeiger und dynamische Speicherverwaltung	115
4.1	Zeiger-Grundlagen	115

4.2	Vorteile von Zeigern	117
4.2.1	Festlegung der Größe von Datenstrukturen zur Laufzeit . . .	117
4.2.2	Größenänderung von Datenstrukturen	117
4.2.3	Gemeinsame Speichernutzung	118
4.3	Wann werden Zeiger verwendet?	119
4.4	Speicherverwaltung	120
4.4.1	Stack und Heap	120
4.4.2	Arbeitsspeicher	124
4.4.3	Lebensdauer	125
4.5	Aufgabenstellungen mit Zeigern	126
4.5.1	Zeichenketten variabler Länge	127
	Aufgabe: Bearbeitung von Zeichenketten variabler Länge	127
4.5.2	Verkettete Listen	139
	Aufgabe: Nachverfolgen einer unbekanntem Zahl von Schülerdatensätzen	139
4.6	Fazit und Ausblick	148
4.7	Übungen	149
5	Klassen	151
5.1	Klassen-Grundlagen	151
5.2	Ziele bei der Verwendung von Klassen	153
5.2.1	Verkapselung	154
5.2.2	Wiederverwendung von Code	155
5.2.3	Zerlegung in Teilaufgaben	155
5.2.4	Information Hiding	156
5.2.5	Verständlichkeit	158
5.2.6	Ausdrucksfähigkeit	159
5.3	Eine einfache Klasse	159
	Aufgabe: Notenliste	160
5.3.1	Grundgerüst einer Klasse	160
5.3.2	Unterstützende Methoden	165
5.4	Klassen mit dynamischen Daten	169
	Aufgabe: Nachverfolgen einer unbekanntem Zahl von Schülerdatensätzen	169
5.4.1	Hinzufügen eines Knotens	172
5.4.2	Umorganisieren einer Liste	175
5.4.3	Destruktor	179
5.4.4	Tiefe Kopien (Deep Copy)	180
5.4.5	Klassen mit dynamischen Daten im Überblick	185

5.5	Fehlervermeidung	186
5.5.1	Fingierte Klassen	187
5.5.2	Monotalente	188
5.6	Übungen	188
6	Rekursion	191
6.1	Grundlagen der Rekursion	191
6.2	Start- und Endrekursion	192
	Aufgabe: Wie viele Papageien?	192
6.2.1	Lösungsweg 1	193
6.2.2	Lösungsweg 2	194
	Aufgabe: Wer ist unser bester Kunde?	196
6.2.3	Lösungsweg 1	198
6.2.4	Lösungsweg 2	199
6.3	Das Hauptkonzept der Rekursion	201
	Aufgabe: Berechnung der Summe eines Arrays von Ganzzahlen	202
6.4	Häufige Fehler	204
6.4.1	Zu viele Parameter	205
6.4.2	Globale Variablen	206
6.5	Rekursion bei dynamischen Datenstrukturen	208
6.5.1	Rekursion und verkettete Listen	208
	Aufgabe: Negative Zahlen in einer einfach verketteten Liste zählen	210
6.5.2	Rekursion und Binärbäume	211
	Aufgabe: Suche nach dem größten Wert in einem Binärbaum	213
6.6	Wrapper-Funktionen	214
	Aufgabe: Anzahl der Blätter eines Binärbaums	214
6.7	Wann wird Rekursion verwendet?	217
6.7.1	Rekursion: Gegenargumente	218
	Aufgabe: Ausgabe einer verketteten Liste	220
	Aufgabe: Ausgabe einer verketteten Liste in umgekehrter Reihenfolge	220
6.8	Übungen	222
7	Wiederverwendung von Code	225
7.1	Sinnvolle und nicht sinnvolle Wiederverwendung von Code	225
7.2	Komponenten	227
7.2.1	Code-Blöcke	227

7.2.2	Algorithmen	227
7.2.3	Entwurfsmuster	228
7.2.4	Abstrakte Datentypen	229
7.2.5	Bibliotheken	230
7.3	Kenntnisse über Komponenten erweitern	230
7.3.1	Forschendes Lernen	231
	Aufgabe: Klassenvorsteher	232
7.3.2	Lernen bei Bedarf	235
	Aufgabe: Effizientes Durchlaufen einer Liste	236
7.4	Auswahl eines Komponententyps	244
7.4.1	Komponentenwahl in der Praxis	246
	Aufgabe: Teilweise Sortierung	246
7.4.2	Vergleich der Ergebnisse	251
7.5	Übungen	251
8	Denken wie ein Programmierer	253
8.1	Das Gesamtkonzept	253
8.1.1	Stärken ausschöpfen, Schwächen lindern	254
8.1.2	Aufbau des Gesamtkonzepts	261
8.2	Beliebige Aufgabenstellungen in Angriff nehmen	262
	Aufgabe: Schummeln beim Galgenmännchen	264
8.2.1	Wie man schummelt	265
8.2.2	Erforderliche Operationen zum Schummeln beim Galgenmännchen	267
8.2.3	Der erste Entwurf	269
8.2.4	Der erste Code	270
8.2.5	Analyse der ersten Ergebnisse	281
8.2.6	Die Kunst des Problemlösens	282
8.3	Programmierkenntnisse weiterentwickeln	283
8.3.1	Neue Programmiersprachen	284
8.3.2	Kenntnisse in bekannten Programmiersprachen erweitern	287
8.3.3	Zusätzliche Bibliotheken	288
8.3.4	Besuchen Sie einen Kurs	289
8.4	Fazit	289
8.5	Übungen	291
	Stichwortverzeichnis	293