

Karl Kurbel

Programmierung und Softwaretechnik



ADDISON-WESLEY

An imprint of Addison Wesley Longman, Inc.

Bonn • Reading, Massachusetts • Menlo Park, California • New York • Harlow, England
Don Mills, Ontario • Sydney • Mexico City • Madrid • Amsterdam

Inhaltsverzeichnis

Vorwort	11
Teil I: Grundlagen für die Entwicklung prozeduraler Programme	15
1 Einführung und Grundbegriffe der Programmentwicklung	17
1.1 Algorithmen, Daten und prozedurale Programme	17
1.2 Einteilung der Programmiersprachen und Programmierparadigmen	19
1.2.1 Programmiersprachen-Generationen	19
1.2.2 Programmierparadigmen	22
1.3 Aufgaben von Programmen	25
1.4 Ein einfaches Rechnermodell	28
1.5 Daten	30
1.5.1 Einfache Datentypen	31
1.5.2 Datenstrukturen und strukturierte Datentypen	33
1.5.2.1 Array	34
1.5.2.2 Record	37
1.5.2.3 Datei	42
1.6 Befehlsarten	44
1.6.1 Ein-/ Ausgabebefehle	45
1.6.2 Zuweisungsbefehle	46
1.6.3 Steuerbefehle	50
1.7 Programmbeispiel	53
1.8 Phasen der Programmentwicklung	54
Aufgaben und Fragen zu Kapitel 1	56
2 Hilfsmittel der Algorithmenentwicklung	59
2.1 Programmablaufpläne	59
2.1.1 Symbole für Programmablaufpläne	59
2.1.2 Beispiele zur Anwendung von Programmablaufplänen ...	61
2.2 Struktogramme	67
2.2.1 Symbole für Struktogramme	67
2.2.2 Beispiele zur Anwendung von Struktogrammen	68
2.3 Entscheidungstabellen	73
2.3.1 Aufbau von Entscheidungsregeln	73
2.3.2 Beispiele zur Anwendung von Entscheidungstabellen	76
Aufgaben und Fragen zu Kapitel 2	78

3 Entwurf und Gestaltung von Programmen	81
3.1 Grundsätze der Programmentwicklung	81
3.1.1 Mängel und Folgen unsystematischer Programmentwicklung	81
3.1.2 Forderungen an Programme	82
3.1.3 Allgemeine Grundsätze für die Gestaltung von Programmen	85
3.2 Entwurfsprinzipien	88
3.2.1 Hierarchische Programmstruktur	88
3.2.2 Top-down-Entwicklung von Algorithmen	90
3.2.2.1 Das Prinzip der stufenweisen Verfeinerung	90
3.2.2.2 Beispiel zur stufenweisen Verfeinerung	91
3.3 Strukturierte Programmierung als Methode zur Programmentwicklung	100
3.3.1 Strukturblöcke	100
3.3.2 Kontrolle des Programmablaufs	102
Aufgaben und Fragen zu Kapitel 3	105
4 Programmstrukturen	107
4.1 Elementare Konstruktionen zur Algorithmenbildung (Steuerkonstrukte)	107
4.1.1 Sequenz	108
4.1.2 Selektion	109
4.1.2.1 Verzweigung	109
4.1.2.2 Fallunterscheidung	110
4.1.3 Repetition	113
4.1.3.1 Schleife mit Vorabprüfung (While-Schleife)	114
4.1.3.2 Zählschleife	116
4.1.3.3 Schleife mit nachträglicher Prüfung (Until-Schleife)	119
4.1.3.4 Cycle-Schleife mit Unterbrechung	120
4.1.4 Beispiel zur Verwendung der Steuerkonstrukte	124
4.2 Programmsegmentierung	127
4.2.1 Begin-Blöcke	127
4.2.2 Unterprogramme	128
4.2.2.1 Parametrisierung von Unterprogrammen	130
4.2.2.2 Arten von Unterprogrammen	131
4.2.2.3 Rekursive Unterprogramme	133
4.2.3 Exkurs: Programmsegmentierung zur Speicherplatzeinsparung	135
Aufgaben und Fragen zu Kapitel 4.	137

5	Spezielle Algorithmen	141
5.1	Verarbeitung von Arrays	141
5.1.1	Eindimensionale Arrays	142
5.1.2	Mehrdimensionale Arrays	149
5.2	Sortieren	153
5.2.1	Sortieren durch Minimumbestimmung	154
5.2.2	Sortieren durch Vertauschen benachbarter Elemente ("Bubble-Sort")	156
5.3	Fortschreibung einer Datei	159
5.3.1	Änderung von Datensätzen	160
5.3.2	Hinzufügen und Löschen von Datensätzen	168
5.4	Gruppenwechsel	173
5.4.1	Einstufiger Gruppenwechsel	174
5.4.2	Mehrstufiger Gruppenwechsel	177
5.4.3	Gruppenwechsel bei der Fortschreibung	185
	Aufgaben und Fragen zu Kapitel 5.	193
6	Beziehungen zwischen Programmen und Daten	195
6.1	Die Stellung des einzelnen Programms in der Gesamtaufgabe	195
6.2	Datenflußpläne	197
	Aufgaben und Fragen zu Kapitel 6.	200
Teil II: Programmierung in einer prozeduralen Programmiersprache		201
7	Die Programmiersprache als Ausdrucksmittel für Algorithmen und Datenbeschreibungen	203
7.1	Die Aufgabe einer Programmiersprache	203
7.2	Anforderungen an eine Programmiersprache und Umsetzung in Pascal	204
7.2.1	Datenvereinbarungen	206
7.2.2	Steuerkonstrukte	211
7.2.3	Ein-/Ausgabebefehle	216
7.2.4	Schrittweise Verfeinerung	219
7.2.5	Selbstdokumentation	222
7.2.6	Beispielprogramme	223

8	Bekannte prozedurale Programmiersprachen	231
8.1	Fortran	231
8.2	Cobol	233
8.3	Basic	234
8.4	Pl/1	235
8.5	Pascal	236
8.6	C	237
8.7	Ada	238
8.8	Weitere Programmiersprachen	239
 Teil III: Von der Modularisierung zur objektorientierten Programmierung		 241
9	Datenabstraktion und Datenkapselung	243
9.1	Modularisierungsprinzipien	243
9.1.1	Abstraktion	244
9.1.1.1	Funktionale Abstraktion	245
9.1.1.2	Datenabstraktion	246
9.1.2	Information Hiding und Datenkapselung	251
9.2	Höhere Datentypen	254
9.2.1	Untertypen	255
9.2.2	Abgeleitete Datentypen	256
9.2.3	Abstrakte Datentypen	258
10	Umsetzung der Datenabstraktion mit den Mitteln einer Programmiersprache	261
10.1	Abstrakte Datenstrukturen	262
10.2	Abstrakte Datentypen	270
10.2.1	Ein Beispiel in Turbo Pascal	272
10.2.2	Information Hiding im Spezifikationsteil	277
10.3	Datenkapseln mit benutzerdefinierten Elementtypen	282
11	Konzepte der objektorientierten Programmierung	287
11.1	Was heißt objektorientierte Entwicklung?	287
11.1.1	Stufen der objektorientierten Entwicklung	288
11.1.2	Prinzipien der objektorientierten Softwareentwicklung	290
11.2	Objekte und Klassen	297
11.3	Kommunikation zwischen Objekten	299
11.4	Beispiel eines objektorientierten Entwurfs	301

12 Umsetzung der objektorientierten Konzepte in einer Programmiersprache	307
12.1 Objektorientierte Programmiersprachen	307
12.2 Objektorientierte Programmierung in Turbo Pascal	308
12.2.1 Abstraktion	308
12.2.2 Kapselung	309
12.2.3 Modularität	311
12.2.4 Hierarchie (Vererbung)	313
12.2.5 Typisierung (Polymorphismus)	316
12.3 Beispiel eines objektorientierten Programmsystems	319
Anhang	
Antworthinweise zu ausgewählten Aufgaben und Fragen	349
Literaturverzeichnis	363
Stichwortverzeichnis	367

Die Entwicklung der objektorientierten Programmiersprachen ist ein Prozess, der sich über Jahrzehnte erstreckt hat. In den 1960er Jahren wurden die ersten objektorientierten Programmiersprachen entwickelt, die sich von den prozeduralen Programmiersprachen unterscheiden. Diese Sprachen waren in der Lage, die Komplexität der Softwareentwicklung zu reduzieren, indem sie die Abstraktion, die Kapselung und die Modularität ermöglichten. Die objektorientierte Programmierung ist heute eine der wichtigsten Paradigmen der Softwareentwicklung. Sie ermöglicht die Entwicklung von Software, die leichter zu verstehen, zu entwickeln und zu warten ist. Die objektorientierte Programmierung ist ein zentraler Bestandteil der modernen Softwareentwicklung. Sie ermöglicht die Entwicklung von Software, die leichter zu verstehen, zu entwickeln und zu warten ist. Die objektorientierte Programmierung ist ein zentraler Bestandteil der modernen Softwareentwicklung. Sie ermöglicht die Entwicklung von Software, die leichter zu verstehen, zu entwickeln und zu warten ist.

Die Evolution der objektorientierten Programmiersprachen ist ein Prozess, der sich über Jahrzehnte erstreckt hat. In den 1960er Jahren wurden die ersten objektorientierten Programmiersprachen entwickelt, die sich von den prozeduralen Programmiersprachen unterscheiden. Diese Sprachen waren in der Lage, die Komplexität der Softwareentwicklung zu reduzieren, indem sie die Abstraktion, die Kapselung und die Modularität ermöglichten. Die objektorientierte Programmierung ist heute eine der wichtigsten Paradigmen der Softwareentwicklung. Sie ermöglicht die Entwicklung von Software, die leichter zu verstehen, zu entwickeln und zu warten ist. Die objektorientierte Programmierung ist ein zentraler Bestandteil der modernen Softwareentwicklung. Sie ermöglicht die Entwicklung von Software, die leichter zu verstehen, zu entwickeln und zu warten ist.