



Inhaltsverzeichnis

1	Einleitung	1
1.1	Die Sprache »C« und ihre Folgen	2
1.2	Drei Schritte zum Glück	3
1.3	Was ist ein Bug?	4
2	Beliebte Fehler	7
2.1	Punkt, Punkt, Komma, Strich	7
2.1.1	Falsche Strichpunkte	7
2.1.2	Komma	9
2.2	Fehlende Klammern	10
2.3	Konstanten	11
2.4	Kommentare	11
2.4.1	Geschachtelte Kommentare	11
2.4.2	Division mit Zeigern	12
2.4.3	Andere Kommentarzeichen	13
2.5	Verwechslung von Zuweisung und Vergleich	13
2.6	Reihenfolge der Ausführung	14
2.6.1	Inkrement/Dekrement	14
2.6.2	UND/ODER-Verknüpfung	15
2.6.3	Parameter-Auswertung	16
2.7	Rechnen in C	16
2.7.1	Overflow/Underflow	16
2.7.2	Über den Umgang mit Gleitkommazahlen	17
2.8	Funktionsaufruf	18
2.8.1	Vergessene Funktionsdeklaration	18
2.8.2	Argumentfehler beim Aufruf	19
2.8.3	call-by-reference	20
2.9	Felder	21
2.9.1	Indizierung	21
2.9.2	Mehrdimensionale Felder	21

2.9.3	Strings	22
2.9.4	Mißverständnis zwischen Zeiger und Feld	22
2.10	Zeiger	23
2.10.1	Mißverständnisse bei Zeigeroperationen	24
2.10.2	Ungültige Zeiger	24
2.11	Zeiger und Arrays	25
2.11.1	Zeiger und Arrays sind nicht dasselbe	25
2.11.2	Zeiger und Arrays sind dasselbe	27
2.11.3	Zeiger und Arrays sind doch nicht dasselbe	28
2.12	Bizarre Syntaxfehler	29
2.12.1	function redefined	29
2.12.2	incompatible types	29
2.13	typedef oder #define	30
2.14	Seiteneffekte in Makros	31
2.14.1	Klammerung	31
2.14.2	Seiteneffekte	31
2.15	Portabilität	32
2.15.1	Die Verwendung von Integer-Variablen	32
2.15.2	Die Integer-Division	33
2.16	Das Jahr 2038	33
2.16.1	Das Jahr 00	33
2.16.2	Die printf-Falle	34
2.16.3	Das Jahr-2038-Problem	35
2.17	Optimierungen	36
2.18	Aufgaben	38
2.19	Zusammenfassung	40
3	Programmierstil	43
3.1	Allgemeine Richtlinien	45
3.2	Verwendete Begriffe	45
3.3	Source Files	47
3.3.1	Größe	47
3.3.2	Modul	47
3.3.3	Header-File	47
3.3.4	Datei-Namen	49
3.4	Kommentare	49
3.5	Formatierung	50
3.6	Datentypen	52
3.6.1	Strukturen	54

3.7	Variablen und Konstanten	54
3.8	Zeiger	58
3.9	Funktionen	58
3.9.1	Bibliotheks-Funktionen	59
3.10	Makros	60
3.11	Operatoren	60
3.11.1	cast	60
3.11.2	Bedingter Operator	61
3.11.3	Logische Operatoren	61
3.11.4	Bitweise Operatoren	62
3.11.5	Shift-Operator	62
3.11.6	Inkrement/Dekrement	62
3.11.7	Zuweisung	62
3.11.8	Komma-Operator	63
3.11.9	Klammerung	63
3.12	Kontrollfluß	64
3.12.1	Bedingungen	64
3.12.2	Die switch-Anweisung	65
3.12.3	If-Anweisung	66
3.12.4	Schleifen	67
3.13	Effizienz	67
3.14	Tools	68
3.15	Aufgaben	68
3.16	Zusammenfassung	69
4	Namenskonventionen	71
4.1	Allgemeine Tips	71
4.1.1	Kurze und prägnante Namen	71
4.1.2	Akkurate Namen	71
4.1.3	Einheitliche Namensgebung	72
4.2	Die ungarische Notation	72
4.2.1	Was verbirgt sich hinter der ungarischen Notation?	73
4.2.2	Probleme	74
4.3	Die schwäbische Notation	74
4.3.1	Orientierung an natürlicher Sprache	75
4.3.2	Groß-/Klein-Schreibung	77
4.3.3	Keine redundanten Benennungen	77
4.3.4	this	78
4.3.5	Lokale Variablen	78
4.3.6	Globale Variablen	80

4.4	Zusammenfassung	80
5	Tips	83
5.1	Prototyp	83
	5.1.1 Strenger Prototyp	83
	5.1.2 Variable Anzahl von Argumenten	84
5.2	Operandentausch	85
5.3	Preprozessor-Symbole	85
5.4	typedef	85
	5.4.1 Sinnlose typedefs	85
	5.4.2 Sinnvolle typedefs	86
5.5	Natürlichsprachliche Kodierung	87
	5.5.1 Positiv denken	87
	5.5.2 Einzeiler (Mini-Patterns)	88
5.6	Keine Überraschungen	89
	5.6.1 Kommentare	90
	5.6.2 Die »VHIT«-Methode	90
	5.6.3 Erst überlegen, dann kodieren	91
5.7	Code-Reading	91
	5.7.1 Umfang	92
	5.7.2 Was bringt Code-Reading?	92
	5.7.3 Was kann man damit vermeiden?	93
	5.7.4 Was gibt es sonst noch dazu zu sagen?	93
	5.7.5 Self-Reading	93
5.8	eXtreme Programming (XP)	94
5.9	Warum man sein Programm nicht »test« nennen sollte ..	95
5.10	Zusammenfassung	95
6	Grund-Prinzipien	97
6.1	Ein-/Ausblenden von Debug-Anweisungen	97
6.2	Anderes Verhalten im »Debug-Mode«	98
6.3	»Low-Level«-Funktionen	99
6.4	Debug Level	100
6.5	Bugs kommentieren	101
6.6	Zusammenfassung	102

7	Asserts	105
7.1	assert	105
7.2	Debug-Level	107
7.3	String-assert	108
7.4	AssertBool	108
7.5	AssertTime	109
7.6	Struktur-asserts	109
7.7	Parameter-Überprüfung	110
7.8	Der Pentium-assert	111
	7.8.1 Pentium Inside?	111
	7.8.2 Ist Ihr Betriebssystem auf den Pentium vorbereitet?	111
7.9	Systemnahe Programmierung	112
7.10	Schlußplädoyer für die asserts	113
7.11	Aufgaben	114
7.12	Zusammenfassung	115
8	DebugPrintfs	117
8.1	Wer war Fibonacci?	117
8.2	Normale printfs	120
8.3	Ein-/Ausblenden von Debug-Informationen	122
8.4	Verbose-Levels	123
8.5	DBUG-Library	125
	8.5.1 Eigenschaften	125
	8.5.2 Ein Fallbeispiel	126
	8.5.3 Steuerung	127
	8.5.4 DBUG oder Debugger?	128
	8.5.5 Weitere Einsatzgebiete	128
8.6	Aufgaben	129
8.7	Zusammenfassung	129
9	Fehlersuche	131
9.1	Trau keinem Debugger	131
	9.1.1 Anderes Verhalten	132
	9.1.2 Guter Debugger – schlechter Debugger	133
	9.1.3 Alternativen	134

9.2	Allgemeine Tips	134
9.2.1	Fehler nur einmal machen	134
9.2.2	Der »Momo«-Effekt	134
9.3	Testen	135
9.3.1	Wann testen?	135
9.3.2	Was testen?	136
9.3.3	Wie testen?	137
9.3.4	Tool-Unterstützung	139
9.3.5	Wer testet?	141
9.4	eXtreme Programming (XP)	142
9.5	Zusammenfassung	142
10	Ein-/Ausgabe	145
10.1	Wie öffne ich eine Datei...	145
10.1.1	...unter DOS?	145
10.1.2	...unter Unix?	146
10.2	Wann schlieÙe ich eine Datei?	146
10.3	Erfolgskontrolle	147
10.4	Binäre Ein-/Ausgabe	147
10.4.1	Hardware-Abhängigkeiten	150
10.5	Wechselseitiges Lesen und Schreiben	150
10.6	Low-/High-Level Ein-/Ausgabe	151
10.7	Aufgaben	153
10.8	Zusammenfassung	154
11	Dynamische Speicherverwaltung	157
11.1	Fehlerquellen	157
11.2	Abhilfe	158
11.3	Realisierung	159
11.3.1	Speicheranforderung (Teil 1)	160
11.3.2	Speicherfreigabe	161
11.3.3	Speicheranforderung (Teil 2)	162
11.4	Einbindung	163
11.5	Verbesserungen	165
11.6	C++	166
11.7	Aufgaben	168
11.8	Zusammenfassung	168

12	Garbage Collection	171
12.1	Was ist Garbage Collection?	171
	12.1.1 Vorteile	171
	12.1.2 Nachteile	172
12.2	Garbage Collection für C	172
	12.2.1 Funktionsweise	172
	12.2.2 Beispiel	172
	12.2.3 GC_malloc() als malloc()-Ersatz	173
	12.2.4 Erfahrungswerte	173
	12.2.5 Einsatzgebiete	174
12.3	Auto-Pointer	174
	12.3.1 Was ist ein Auto-Pointer?	174
	12.3.2 Wieso auto_ptr?	175
	12.3.3 Was ist ein SmartPointer?	176
	12.3.4 Ein einfaches Beispiel	176
	12.3.5 Ein ausführlicheres Beispiel	176
	12.3.6 Nachteile von auto_ptr	180
12.4	Zusammenfassung	181
13	ANSI-C mit K&R-C-Compiler	183
13.1	Kann mein Compiler ANSI-C?	183
13.2	Prototyping	184
13.3	Funktions-Definition... ..	186
	13.3.1 ... in der guten alten K&R-Zeit	186
	13.3.2 ... in der ANSI-Neuzeit	187
	13.3.3 Mischen von ANSI-Prototyp mit K&R	187
	13.3.4 ... für beide Welten	188
13.4	Variable Parameter-Liste	190
13.5	Neue Schlüsselwörter	191
13.6	Vorbelegte Preprozessor-Namen	192
13.7	Vor- und Nachteile der Lösung	192
	13.7.1 Vorteile	192
	13.7.2 Nachteile	192
13.8	C++-Unterstützung	193
13.9	Protoize	193
13.10	Aufgaben	193
13.11	Zusammenfassung	194

14	Tools	197
14.1	Entwicklungs-Umgebungen	198
14.2	GUI-Builder	199
14.3	CAST-Tools	199
14.4	Traue keinem Compiler	200
	14.4.1 Lint	200
	14.4.2 LCLint	202
	14.4.3 «CLINT»	203
	14.4.4 ProLint - ein Erfahrungsbericht	203
14.5	Weitere Code-Checker	207
	14.5.1 Insure++	207
	14.5.2 Purify	213
	14.5.3 CodeCenter/ObjectCenter	214
14.6	Q-Tools	214
14.7	Freeware/Shareware	215
	14.7.1 Entwicklungsumgebung	215
	14.7.2 Compiler	216
	14.7.3 C++-Compiler	216
	14.7.4 Debugger	217
14.8	Bibliotheken	217
	14.8.1 Debug-Bibliothek	217
	14.8.2 Memory Profiler	217
	14.8.3 Libretto	218
14.9	Unix-Tools	218
14.10	Zusammenfassung	219
15	Einstellungssache	221
15.1	Allgemeine Hinweise	221
	15.1.1 Spracherweiterungen	221
	15.1.2 Warn-Level	221
	15.1.3 Optimierungsschalter	221
	15.1.4 Handbücher	222
15.2	GNU-Compiler	222
	15.2.1 Warnungen	223
	15.2.2 Optimierung	224
	15.2.3 Vordefinierte Preprozessor-Symbole	225
	15.2.4 Verwendete Header-Dateien	225
15.3	Visual C++	225
	15.3.1 Editor-Einstellungen	225

15.3.2	Projekt-Einstellungen	226
15.3.3	Fehlermeldungen	226
15.3.4	STL-Bibliothek	227
15.4	Borland Compiler	227
15.4.1	IDE-Einstellungen	227
15.4.2	Projekt-Einstellungen	227
15.4.3	Warnungen	228
15.4.4	Fehlermeldungen	228
15.5	Lint-Optionen	228
15.5.1	Portabilität	229
15.5.2	Spezielle Kommentare	229
15.5.3	Unused Return Value	230
15.5.4	Wie bringe ich lint sonst noch zum Schweigen?	231
15.6	Aufgaben	231
15.7	Zusammenfassung	231
16	Versionskontrolle	233
16.1	Grundidee	234
16.1.1	Versionierung	234
16.1.2	Sicherungen	234
16.2	Kurze Einführung in SCCS	235
16.3	Problemstellung	235
16.3.1	Lösung 1	236
16.3.2	Lösung 2	236
16.3.3	Erweiterung	237
16.4	RCS	237
16.4.1	RCS-Lösung	238
16.5	CVS	238
16.6	Testversionen	238
16.7	Zusammenfassung	242
	Anhang	245
A	Lösungen...	245
A.1	... zu Kapitel 2	245
A.2	... zu Kapitel 3	246
A.3	... zu Kapitel 7	249
A.4	... zu Kapitel 8	249
A.5	... zu Kapitel 10	250
A.6	... zu Kapitel 11	250

A.7	... zu Kapitel 13	251
A.8	... zu Kapitel 15	252
B	Übungsbeispiele	253
B.1	strncpy.c	253
B.2	That's »life«	254
B.3	Die fibo-Reihe	258
B.4	karte*.[ch]	261
B.5	q.c	264
C	Listings	267
C.1	odebug.h	267
C.2	ansi.h	272
C.3	libonew.a	275
D	DBUG-Library	285
D.1	Manual	285
D.2	Bezugsquellen	292
D.3	Plattformen	293
E	Literaturverzeichnis	295
E.1	Bücher	295
E.2	Internet	300
F	Die beiliegende CD	303
F.1	Inhalt	303
F.2	Spezielle Dateien	304
G	Glossary	305
G.1	Allgemein	305
G.2	Abkürzungen	309
	Index	311