

Inhalt

Vorwort	21
Vorwort zur 2. Auflage	22
Vorwort des Gutachters	25
1 Einstieg in C	27
1.1 Übersicht zu C	27
1.2 Der ANSI C-Standard	28
1.2.1 Der Vorteil des ANSI C-Standards	30
1.3 Der POSIX-Standard	30
1.4 Vor- und Nachteile der Programmiersprache C	31
1.5 C in diesem Buch	32
1.6 Was benötige ich für C?	33
1.6.1 Texteditor	33
1.6.2 Compiler	33
1.6.3 All-in-one – die Entwicklungsumgebung	34
1.7 Welcher Compiler und welches Betriebssystem?	35
2 Das erste Programm	37
2.1 Der beste Lernerfolg	37
2.2 »Hallo Welt« in C	37
2.3 Analyse des Programms	39
3 Zeichensätze	43
3.1 Basic-Zeichensatz	43
3.2 Ausführungszeichensatz (Steuerzeichen)	43

4	Kommentare in C	47
4.1	Wann sind Kommentare sinnvoll?	47
4.2	Welche Kommentar-Schreibweise: // oder /* */	48
5	Formatierte Eingabe mit scanf()	51
5.1	Der Adressoperator »&«	52
5.2	Probleme und deren Behandlung mit scanf()	54
5.2.1	Möglichkeit 1	55
5.2.2	Möglichkeit 2	55
5.2.3	Möglichkeit 3	56
5.3	Format prüfen	57
5.4	Zusammenfassung scanf()	58
6	Formatierte Ausgabe mit printf	61
7	Elementare Datentypen	63
7.1	Der Datentyp int (Integer)	63
7.2	Variablen deklarieren	64
7.2.1	Erlaubte Bezeichner	67
7.3	C versus C++ bei der Deklaration von Variablen	68
7.4	Der Datentyp long	68
7.5	Der Datentyp short	69
7.6	Die Gleitpunkttypen float und double	70
7.6.1	Gleitpunkttypen im Detail	72
7.6.2	float im Detail	72
7.6.3	double im Detail	73
7.6.4	long double	73
7.6.5	Einiges zu n-stelliger Genauigkeit	74
7.7	Numerische Gleitpunktprobleme	76
7.8	Der Datentyp char	77
7.9	Nationale contra internationale Zeichensätze	82
7.10	Vorzeichenlos und vorzeichenbehaftet	83
7.11	Limits für Ganzzahl- und Gleitpunktdatentypen	85
7.12	Konstanten	88
7.12.1	Ganzzahlkonstanten	88
7.12.2	Gleitpunktkonstanten	89
7.12.3	Zeichenkonstanten	89
7.12.4	String-Literale (Stringkonstante)	89
7.13	Umwandlungsvorgaben für formatierte Ein-/Ausgabe	90

8 Operatoren 97

8.1	Exkurs zu Operatoren	97
8.2	Arithmetische Operatoren	98
8.2.1	Dividieren von Ganzzahlen	99
8.3	Erweiterte Darstellung arithmetischer Operatoren	101
8.4	Inkrement- und Dekrement-Operatoren	102
8.5	Bit-Operatoren	103
8.5.1	Bitweises UND	104
8.5.2	Bitweises ODER	106
8.5.3	Bitweise XOR	107
8.5.4	Bitweises Komplement	108
8.5.5	Linksverschiebung	108
8.5.6	Rechtsverschiebung	109
8.5.7	Rezept für Fortgeschrittene	109
8.6	sizeof-Operator	111
8.6.1	C versus C++	113

9 Typenumwandlung 115

10 Kontrollstrukturen 117

10.1	Verzweigungen mit der if-Bedingung	117
10.1.1	Anweisungsblock	117
10.2	Die Verzweigung mit else if	122
10.3	Die Verzweigung mit else	123
10.4	Der !-Operator (logischer Operator)	127
10.5	Logisches UND (&&) – Logisches ODER ()	129
10.6	Bedingungsoperator ?:	132
10.7	Fallunterscheidung: die switch-Verzweigung	134
10.7.1	default	137
10.8	Die while-Schleife	138
10.8.1	Endlosschleife (while)	140
10.8.2	Fehlervermeidung bei while-Schleifen	141
10.9	Die do while-Schleife	142
10.10	Die for-Schleife	146
10.10.1	Beispiele für eine for-Schleife	149
10.11	Kontrollierte Sprünge	153
10.11.1	continue	153
10.11.2	break	155
10.12	Direkte Sprünge mit goto	155

10.13	Notationsstil	156
10.13.1	K&R-Stil	157
10.13.2	Whitesmith-Stil	157
10.13.3	Allman-Stil	157
10.13.4	GNU EMACS-Stil	157
10.13.5	Der Stil des Autors ;) (K&R-like)	157

11 Funktionen 159

11.1	Was sind Funktionen?	159
11.2	Wozu Funktionen?	159
11.3	Definition von Funktionen	159
11.4	Funktionsaufruf	160
11.5	Funktionsdeklaration	162
11.6	Lokale Variablen	163
11.7	Globale Variablen	166
11.8	Statische Variablen	167
11.9	Schlüsselworte für Variablen – Speicherklassen	168
11.9.1	auto	169
11.9.2	extern	169
11.9.3	register	169
11.9.4	static	170
11.10	Typ-Qualifizierer	170
11.10.1	volatile	170
11.10.2	const	170
11.11	Geltungsbereich von Variablen	171
11.12	Speicherklassen-Spezifizierer für Funktionen	172
11.12.1	extern	172
11.12.2	static	172
11.12.3	volatile	173
11.13	Datenaustausch zwischen Funktionen	173
11.14	Wertübergabe an Funktionen (call-by-value)	174
11.15	Rückgabewert von Funktionen	178
11.16	Die Hauptfunktion main()	179
11.17	Rückgabewert beim Beenden eines Programms	181
11.18	Funktionen der Laufzeitbibliothek	182
11.19	Getrenntes Compilieren von Quelldateien	183
11.20	Rekursive Funktionen	185
11.20.1	Exkurs: Stack	186
11.20.2	Rekursionen und der Stack	186
11.20.3	Fakultät	191
11.20.4	Fibonacci-Zahlen	192
11.20.5	Größter gemeinsamer Teiler (GGT)	193

12 Präprozessor-Direktiven 199

12.1	Einkopieren von Dateien mittels #include	199
12.2	Makros und Konstanten – #define	203
12.2.1	Symbolische Konstanten mit #define	203
12.2.2	Makros mit #define	208
12.3	Bedingte Kompilierung	211
12.4	Vordefinierte Präprozessor-Direktiven (ANSI C)	217
12.5	Ersetzung eines Makroparameters durch einen String	219
12.6	#undef – Makronamen wieder aufheben	220
12.7	Ausgeben von Fehlermeldungen – #error	221
12.8	#pragma	222

13 Arrays 223

13.1	Arrays deklarieren	223
13.2	Initialisierung und Zugriff auf Arrays	224
13.2.1	Gültigkeitsbereich von Arrays	229
13.3	Arrays vergleichen	231
13.4	Anzahl der Elemente eines Arrays ermitteln	233
13.5	Übergabe von Arrays an Funktionen	234
13.6	Arrays aus Funktionen zurückgeben	236
13.7	Programmbeispiel zu den Arrays	237
13.8	Einlesen von Array-Werten	241
13.9	Mehrdimensionale Arrays	242
13.10	Mehrdimensionale Arrays initialisieren	242
13.10.1	Tic Tac Toe	250
13.10.2	Dreidimensionale Arrays	255
13.11	Übergabe von zwei- bzw. mehrdimensionalen Arrays an Funktionen ...	256
13.12	Arrays in Tabellenkalkulation einlesen (*.CSV-Dateien)	258
13.13	Strings/Zeichenketten (char Array)	259
13.13.1	Vom String zur Binärzahl	263
13.14	Einlesen von Strings	266
13.15	Standard-Bibliothek <string.h>	268
13.15.1	strcat() – Strings aneinander hängen	269
13.15.2	strchr() – ein Zeichen im String suchen	270
13.15.3	strcmp() – Strings vergleichen	270
13.15.4	strcpy() – einen String kopieren	271
13.15.5	strcspn() – einen Teilstring ermitteln	272
13.15.6	strlen() – Länge eines Strings ermitteln	272
13.15.7	strncat() – String mit n Zeichen aneinander hängen	273

13.15.8	strncmp() – n Zeichen von zwei Strings miteinander vergleichen	274
13.15.9	strncpy() – String mit n Zeichen kopieren	275
13.15.10	strpbrk() – Auftreten bestimmter Zeichen suchen	276
13.15.11	strrchr() – das letzte Auftreten eines bestimmten Zeichens im String suchen	276
13.15.12	strspn() – erstes Auftreten eines Zeichens, das nicht vorkommt	277
13.15.13	strstr() – String nach Auftreten eines Teilstrings durchsuchen	277
13.15.14	strtok() – String anhand bestimmter Zeichen zerlegen	278

14 Zeiger (Pointer) 281

14.1	Zeiger deklarieren	281
14.2	Zeiger initialisieren	282
14.2.1	Speichergröße von Zeigern	294
14.3	Zeigerarithmetik	295
14.4	Zeiger, die auf andere Zeiger verweisen	295
14.4.1	Subtraktion zweier Zeiger	297
14.5	Typensicherung bei der Dereferenzierung	298
14.6	Zeiger als Funktionsparameter (call-by-reference)	298
14.6.1	Zeiger als Rückgabewert	302
14.7	Array und Zeiger	305
14.8	Zeiger auf Strings	313
14.8.1	Zeiger auf konstante Objekte (Read-only-Zeiger)	314
14.9	Zeiger auf Zeiger und Stringtabellen	315
14.9.1	Stringtabellen	316
14.10	Zeiger auf Funktionen	325
14.11	void-Zeiger	331
14.12	Äquivalenz zwischen Zeigern und Arrays	334

15 Kommandozeilenargumente 339

15.1	Argumente an die Hauptfunktion	339
15.2	Optionen (Schalter) aus der Kommandozeile auswerten	345

16 Dynamische Speicherverwaltung 351

16.1	Das Speicherkonzept	351
16.2	Speicherallozierung mit malloc()	352

16.3	Die Mystery von NULL	356
16.3.1	NULL für Fortgeschrittene	356
16.3.2	Was jetzt – NULL, 0 oder \0 ... ?	358
16.3.3	Zusammengefasst	358
16.4	Speicherreservierung und ihre Probleme	359
16.5	free() – Speicher wieder freigeben	360
16.6	Die Freispeicherverwaltung	363
16.6.1	Prozessinterne Freispeicherverwaltung	365
16.7	Dynamisches Array	367
16.8	Speicher dynamisch reservieren mit realloc und calloc	371
16.9	Speicher vom Stack anfordern mit alloca (nicht ANSI C)	375
16.10	free – Speicher wieder freigeben	375
16.11	Zweidimensionale dynamische Arrays	376
16.12	Wenn die Speicherallozierung fehlschlägt	379
16.12.1	Speicheranforderung reduzieren	380
16.12.2	Speicheranforderungen aufteilen	381
16.12.3	Einen Puffer konstanter Größe verwenden	382
16.12.4	Zwischenspeichern auf Festplatte vor der Allokation	383
16.12.5	Nur so viel Speicher anfordern wie nötig	383

17 Strukturen 385

17.1	Struktur deklarieren	385
17.2	Initialisierung und Zugriff auf Strukturen	386
17.3	Strukturen als Wertübergabe an eine Funktion	391
17.4	Strukturen als Rückgabewert einer Funktion	393
17.5	Strukturen vergleichen	395
17.6	Arrays von Strukturen	397
17.7	Strukturen in Strukturen (Nested Structures)	404
17.8	Kurze Zusammenfassung zu den Strukturen	415
17.9	Union	416
17.10	Aufzählungstyp enum	421
17.11	Typendefinition mit typedef	424
17.12	Attribute von Strukturen verändern (nicht ANSI C)	428
17.13	Bitfelder	431
17.14	Das offsetof-Makro	438

18 Ein-/Ausgabe-Funktionen 441

18.1	Was ist eine Datei?	441
18.2	Formatierte und unformatierte Ein-/Ausgabe	441

18.3	Streams	442
18.4	Höhere Ein-/Ausgabe-Funktionen	442
18.5	Datei (Stream) öffnen – fopen	443
18.5.1	Modus für fopen()	446
18.5.2	Maximale Anzahl geöffneter Dateien – FOPEN_MAX	448
18.6	Zeichenweise Lesen und Schreiben – getchar und putchar	449
18.6.1	Ein etwas portableres getch()	451
18.7	Zeichenweise Lesen und Schreiben – putc/fputc und getc/fgetc	453
18.8	Datei (Stream) schließen – fclose	459
18.9	Formatiertes Einlesen/Ausgeben von Streams mit fprintf und fscanf	462
18.10	Standard-Streams in C	467
18.10.1	Standard-Streams umleiten	468
18.11	Fehlerbehandlung von Streams – feof, ferror und clearerr	470
18.12	Gelesenes Zeichen in die Eingabe zurück-schieben – ungetc	472
18.13	(Tastatur-)Puffer leeren – fflush	474
18.13.1	Pufferung	475
18.14	Stream positionieren – fseek, rewind und ftell	476
18.15	Stream positionieren – fsetpos, fgetpos	480
18.16	Zeilenweise Ein-/Ausgabe von Streams	481
18.16.1	Zeilenweise Lesen mit gets/fgets	481
18.16.2	Zeilenweise Schreiben mit puts/fputs	485
18.16.3	Zeilenweise Einlesen vom Stream mit getline() (nicht ANSI C)	486
18.16.4	Rezepte für zeilenweises Einlesen und Ausgeben	488
18.17	Blockweise Lesen und Schreiben – fread und fwrite	496
18.17.1	Blockweises Lesen – fread()	497
18.17.2	Blockweises Schreiben – fwrite()	498
18.17.3	Big-Endian und Little-Endian	503
18.18	Datei (Stream) erneut öffnen – freopen	506
18.19	Datei löschen oder umbenennen – remove und rename	507
18.19.1	remove()	507
18.19.2	rename()	509
18.20	Pufferung einstellen – setbuf und setvbuf	510
18.21	Temporäre Dateien erzeugen – tmpfile und tmpnam	516
18.21.1	mkstemp() – Sichere Alternative für Linux/UNIX (nicht ANSI C)	520
18.22	Fehlerausgabe mit stderr und perror	521
18.23	Formatiert in einem String schreiben und formatiert aus einem String lesen – sscanf und sprintf	525
18.24	Fortgeschrittenes Thema	529
18.25	Low-Level-Datei-I/O-Funktionen (nicht ANSI C)	537
18.26	Datei öffnen – open	538
18.27	Datei schließen – close	545

18.28	Datei erzeugen – creat	546
18.29	Schreiben und Lesen – write und read	547
18.30	File-Deskriptor positionieren – lseek	558
18.31	File-Deskriptor von einem Stream – fileno	559
18.32	Stream von File-Deskriptor – fdopen	560

19 Attribute von Dateien und Arbeiten mit Verzeichnissen (nicht ANSI C) 563

19.1	Attribute einer Datei ermitteln – stat()	563
19.1.1	stat() – st_mode	564
19.1.2	stat() – st_size	570
19.1.3	stat() – st_atime, st_mtime und st_ctime	571
19.1.4	stat() – st_gid und st_uid	576
19.1.5	stat() – st_nlink, st_ino	577
19.1.6	stat() – st_dev, st_rdev	577
19.2	Prüfen des Zugriffsrechts – access	580
19.3	Verzeichnis-Funktionen	583
19.3.1	Verzeichnis erstellen, löschen und wechseln – mkdir, rmdir und chdir	583
19.3.2	Wechseln in das Arbeitsverzeichnis – getcwd	589
19.3.3	Verzeichnisse öffnen, lesen und schließen – opendir, readdir und closedir	591

20 Arbeiten mit variablen langen Argumentlisten – <stdarg.h> 597

20.1	Makros in <stdarg.h> – va_list, va_arg, va_start und va_end	597
20.2	Argumentliste am Anfang oder Ende kennzeichnen	598
20.3	vprintf, vsprintf und vfprintf	602

21 Zeitroutinen 607

21.1	Die Headerdatei <time.h>	607
21.1.1	Konstanten in der Headerdatei <time.h>	608
21.1.2	Datums- und Zeitfunktionen in <time.h>	608
21.2	Laufzeitmessung (Profiling)	618

22 Weitere Headerdateien und ihre Funktionen (ANSI C) 621

22.1	<assert.h> – Testmöglichkeiten und Fehlersuche	621
22.2	<ctype.h> – Zeichenklassifizierung und Umwandlung	623
22.3	Mathematische Funktionen – <math.h>	626
22.4	<stdlib.h>	628
22.4.1	Programmbeendigung – exit(), _exit(), atexit() und abort()	628
22.4.2	Konvertieren von Strings in numerische Werte	631
22.4.3	Bessere Alternative – Konvertieren von Strings in numerische Werte	633
22.4.4	Zufallszahlen	638
22.4.5	Absolutwerte, Quotient und Rest von Divisionen	639
22.4.6	Suchen und Sortieren – qsort() und bsearch()	641
22.4.7	system()	644
22.5	<locale.h> – Länderspezifische Eigenheiten	645
22.6	<setjmp.h>	648
22.7	<signal.h>	653
22.8	<string.h> – Die mem...-Funktionen zur Speicheranipulation	658
22.8.1	memchr() – Suche nach einzelnen Zeichen	658
22.8.2	memcmp() – Bestimmte Anzahl von Bytes vergleichen	659
22.8.3	memcpy() – Bestimmte Anzahl von Bytes kopieren	660
22.8.4	memmove() – Bestimmte Anzahl von Bytes kopieren	660
22.8.5	memset() – Speicherbereich mit bestimmten Zeichen auffüllen	661
22.9	Erweiterter ANSI C-Standard (ANSI C99)	662
22.9.1	Neue elementare Datentypen	662
22.9.2	<stdint.h> – Ganzzahlige Typen mit vorgegebener Breite	662
22.9.3	Komplexe Gleitpunkttypen	663
22.9.4	<iso646.h> – Symbolische Konstanten für Operatoren	663
22.9.5	Deklaration von Bezeichnern	664
22.9.6	inline-Funktionen	664
22.9.7	Vordefinierte Makros	665
22.9.8	<math.h> – Neue Funktionen	666
22.9.9	<wchar.h> – (NA1)	668
22.9.10	<wctype.h> (NA1)	668
22.9.11	<fenv.h> – Kontrolle der Gleitpunktzahlen-Umgebung	669
22.9.12	<inttypes.h> – Für genauere Integer-Typen	669
22.9.13	<tgmath.h> – Typengenerische Mathematik-Funktionen	669
22.9.14	Zusammenfassung	670

23 Dynamische Datenstrukturen 671

23.1	Lineare Listen (einfach verkettete Listen)	671
23.1.1	Erstes Element der Liste löschen	678

23.1.2	Beliebiges Element in der Liste löschen	680
23.1.3	Elemente der Liste ausgeben	683
23.1.4	Eine vollständige Liste auf einmal löschen	689
23.1.5	Element in die Liste einfügen	691
23.2	Doppelt verkettete Listen	698
23.3	Stacks nach dem LIFO (Last-in-First-out)-Prinzip	715
23.4	Queues nach dem FIFO-Prinzip	737

24 Algorithmen 747

24.1	Was sind Algorithmen?	747
24.2	Wie setze ich Algorithmen ein?	747
24.3	Sortieralgorithmen	748
24.3.1	Selektion Sort – Sortieren durch Auswählen	748
24.3.2	Insertion Sort	750
24.3.3	Bubble Sort	752
24.3.4	Shellsort	754
24.3.5	Quicksort	758
24.3.6	qsort()	764
24.3.7	Zusammenfassung der Sortieralgorithmen	766
24.4	Suchalgorithmen – Grundlage zur Suche	774
24.4.1	Lineare Suche	775
24.4.2	Binäre Suche	777
24.4.3	Binäre (Such-)Bäume	780
24.4.4	Elemente im binären Baum einordnen	782
24.4.5	Binäre Bäume traversieren	787
24.4.6	Löschen eines Elements im binären Baum	788
24.4.7	Ein binärer Suchbaum in der Praxis	791
24.4.8	Binäre Suchbäume mit Eltern-Zeiger und Threads	801
24.4.9	Ausgeglichene Binärbäume	802
24.4.10	Algorithmen für ausgeglichene Bäume – eine Übersicht	803
24.5	Hashing (Zerhacken)	804
24.5.1	Wann wird Hashing verwendet?	804
24.5.2	Was ist für das Hashing erforderlich?	804
24.5.3	Hash-Funktion	809
24.5.4	Hashing mit direkter Adressierung	814
24.5.5	Vergleich von Hashing mit binären Bäumen	814
24.6	String-Matching	815
24.6.1	Brute-Force-Algorithmus	816
24.6.2	Der Algorithmus von Knuth/Morris/Pratt (KMP)	819
24.6.3	Weitere String-Matching-Algorithmen	826
24.7	Pattern Matching (reguläre Ausdrücke)	826
24.8	Backtracking	833
24.8.1	Der Weg durch den Irrgarten	833
24.8.2	Das 8-Dame-Problem	847

25 Sicheres Programmieren 857

25.1	Buffer Overflow (Speicherüberlauf)	858
25.1.1	Speicherverwaltung von Programmen	859
25.1.2	Der Stack-Frame	860
25.1.3	Rücksprungadresse manipulieren	861
25.1.4	Gegenmaßnahmen zum Buffer Overflow während der Programmerstellung	868
25.1.5	Gegenmaßnahmen zum Buffer Overflow, wenn das Programm fertig ist	871
25.1.6	Programme und Tools zum Buffer Overflow	874
25.1.7	Ausblick	875
25.2	Memory Leaks (Speicherlecks)	876
25.2.1	Bibliotheken und Tools zu Memory Leaks	880
25.3	Tipps zu Sicherheitsproblemen	881

26 CGI mit C 883

26.1	Was ist CGI?	883
26.2	Vorteile von CGIs in C	883
26.3	Andere Techniken der Webprogrammierung	884
26.4	Das dreistufige Webanwendungsdesign	885
26.4.1	Darstellungsschicht	885
26.4.2	Verarbeitungsschicht	886
26.4.3	Speicherschicht	886
26.5	Clientseitige Programmierung	887
26.5.1	JavaScript	887
26.5.2	Java-Applets	887
26.6	Serverseitige Programmierung	887
26.7	Der Webserver	888
26.7.1	Das Client/Server-Modell des Internets	888
26.7.2	Serverimplementierung	889
26.7.3	Hosting-Services	890
26.7.4	Schlüsselfertige Lösung	890
26.7.5	Weitere Möglichkeiten	891
26.7.6	Apache	891
26.8	Das HTTP-Protokoll	901
26.8.1	Web-Protokolle	901
26.8.2	Wozu Protokolle?	901
26.8.3	Was ist ein Protokoll?	901
26.8.4	Normen für die Netzwerktechnik	902
26.8.5	Das OSI-Schichtenmodell	902
26.8.6	Die Elemente einer URL	903
26.8.7	Client-Anfrage – HTTP Request (Browser-Request)	905
26.8.8	Serverantwort (Server-Response)	908
26.8.9	Zusammenfassung	911

26.9	Das Common Gateway Interface (CGI)	911
26.9.1	Filehandles	911
26.9.2	CGI-Umgebungsvariablen	912
26.9.3	CGI-Ausgabe	917
26.10	HTML-Formulare	920
26.10.1	Die Tags und ihre Bedeutung	920
26.11	CGI-Eingabe	927
26.11.1	Die Anfrage des Clients an den Server	927
26.11.2	Eingabe parsen	931
26.12	Ein Gästebuch	936
26.12.1	Das HTML-Formular (guestbook.html)	936
26.12.2	Das CGI-Programm (auswert.cgi)	938
26.12.3	Das HTML-Gästebuch (gaeste.html)	947
26.12.4	Das Beispiel ausführen	947
26.13	Ausblick	949

27 MySQL und C 951

27.1	Aufbau eines Datenbanksystems	951
27.1.1	Warum wurde ein Datenbanksystem (DBS) entwickelt?	951
27.1.2	Das Datenbank-Management-System (DBMS)	952
27.1.3	Relationale Datenbank	954
27.1.4	Eigene Clients mit C für SQL mit der ODBC-API entwickeln	955
27.2	MySQL installieren	956
27.2.1	Linux	956
27.2.2	Windows	956
27.2.3	Den Client mysql starten	957
27.3	Crashkurs (My)SQL	958
27.3.1	Was ist SQL?	958
27.3.2	Die Datentypen von (My)SQL	958
27.3.3	Eine Datenbank erzeugen	960
27.3.4	Eine Datenbank löschen	961
27.3.5	Datenbank wechseln	962
27.3.6	Eine Tabelle erstellen	962
27.3.7	Die Tabelle anzeigen	963
27.3.8	Tabellendefinition überprüfen	963
27.3.9	Tabelle löschen	964
27.3.10	Struktur einer Tabelle ändern	964
27.3.11	Datensätze eingeben	965
27.3.12	Datensätze auswählen	966
27.3.13	Ein fortgeschrittenes Szenario	967
27.3.14	Datensatz löschen	968
27.3.15	Datensatz ändern	968
27.3.16	Zugriffsrechte in MySQL	968
27.3.17	Übersicht über einige SQL-Kommandos	970
27.4	Die MySQL C-API	971
27.4.1	Grundlagen zur Programmierung eines MySQL-Clients	971

27.4.2	Client-Programm mit dem gcc- unter Linux und dem Cygwin gcc-Compiler unter Windows	973
27.4.3	MySQL Client-Programme mit dem VC++ Compiler und dem Borland Freeware Compiler	973
27.4.4	Troubleshooting	975
27.4.5	Das erste Client-Programm – Verbindung mit dem MySQL-Server herstellen	975
27.4.6	MySQL-Kommandozeilen-Optionen	980
27.4.7	Anfrage an den Server	983
27.5	MySQL und C mit CGI	1003
27.5.1	HTML-Eingabeformular	1003
27.5.2	CGI-Anwendung add_db.cgi	1004
27.5.3	CGI-Anwendung search_db.cgi	1013
27.6	Funktionsübersicht	1022
27.7	Datentypenübersicht der C-API	1025

28 Netzwerkprogrammierung und Cross-Plattform-Entwicklung **1027**

28.1	Begriffe zur Netzwerktechnik	1027
28.1.1	IP-Nummern	1028
28.1.2	Portnummer	1029
28.1.3	Host- und Domainname	1030
28.1.4	Nameserver	1030
28.1.5	Das IP-Protokoll	1031
28.1.6	TCP und UDP	1031
28.1.7	Was sind Sockets?	1032
28.2	Headerdateien zur Socketprogrammierung	1033
28.2.1	Linux/UNIX	1033
28.2.2	Windows	1033
28.3	Client-/Server-Prinzip	1036
28.3.1	Loopback-Interface	1036
28.4	Erstellen einer Client-Anwendung	1037
28.4.1	socket() – Erzeugen eines Kommunikationsendpunktes	1037
28.4.2	connect() – Client stellt Verbindung zum Server her	1039
28.4.3	Senden und Empfangen von Daten	1044
28.4.4	close(), closesocket()	1047
28.5	Erstellen einer Server-Anwendung	1048
28.5.1	bind() – Festlegen einer Adresse aus dem Namensraum	1048
28.5.2	listen() – Warteschlange für eingehende Verbindungen einrichten	1050
28.5.3	accept() und die Serverhauptschleife	1051
28.6	(Cross-Plattform)TCP-Echo-Server	1054
28.6.1	Der Client	1054
28.6.2	Der Server	1057

28.7	Cross-Plattform-Development	1061
28.7.1	Abstraktion Layer	1062
28.7.2	Headerdatei Linux/UNIX	1062
28.7.3	Linux/UNIX-Quelldatei	1063
28.7.4	Headerdatei MS-Windows	1067
28.7.5	Windows-Quelldatei	1068
28.7.6	All together – die main-Funktionen	1072
28.7.7	Ein UDP-Beispiel	1075
28.7.8	Mehrere Clients gleichzeitig behandeln	1078
28.8	Weitere Anmerkungen zur Netzwerkprogrammierung	1086
28.8.1	Das Datenformat	1086
28.8.2	Der Puffer	1087
28.8.3	Portabilität	1088
28.8.4	Von IPv4 nach IPv6	1088
28.8.5	RFC-Dokumente (Request for Comments)	1090
28.8.6	Sicherheit	1090

29 Wie geht's jetzt weiter? 1091

29.1	GUI-Programmierung – Grafische Oberflächen	1091
29.1.1	Low-Level-Grafikprogrammierung	1092
29.1.2	High-Level-Grafikprogrammierung	1092
29.1.3	Multimedia-Grafikprogrammierung	1093

A Anhang 1095

A.1	Rangfolge der Operatoren	1095
A.2	ASCII-Code-Tabelle	1097
A.3	Reservierte Schlüsselwörter in C	1098
A.4	Standard-Headerdateien der ANSI C-Bibliothek	1098
A.5	Weiterführende Links	1098
A.6	Weiterführende Literatur	1098
A.6.1	Bücher zum ANSI C-Standard	1099
A.6.2	Algorithmen und Datenstrukturen	1099
A.6.3	HTML, CGI-Webprogrammierung	1100
A.6.4	Linux/UNIX	1100
A.6.5	MySQL	1101

Index 1103