

Auf einen Blick

Vorwort	21
Vorwort des Gutachters	25
1 Einstieg in C	27
2 Das erste Programm	37
3 Zeichensätze	43
4 Kommentare in C	47
5 Formatierte Eingabe mit scanf()	51
6 Formatierte Ausgabe mit printf	61
7 Elementare Datentypen	63
8 Operatoren	97
9 Typenumwandlung	115
10 Kontrollstrukturen	117
11 Funktionen	159
12 Präprozessor-Direktiven	199
13 Arrays	223
14 Zeiger (Pointer)	281
15 Kommandozeilenargumente	339
16 Dynamische Speicherverwaltung	351
17 Strukturen	385
18 Ein-/Ausgabe-Funktionen	441
19 Attribute von Dateien und Arbeiten mit Verzeichnissen (nicht ANSI C)	563
20 Arbeiten mit variablen langen Argumentlisten – <stdarg.h> ...	597
21 Zeitroutinen	607
22 Weitere Headerdateien und ihre Funktionen (ANSI C)	621
23 Dynamische Datenstrukturen	671
24 Algorithmen	747
25 Sicheres Programmieren	857
26 CGI mit C	883
27 MySQL und C	951
28 Netzwerkprogrammierung und Cross-Plattform -Entwicklung	1027
29 Wie geht's jetzt weiter?	1091
A Anhang	1095
Index	1103

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 Allozierung	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 perror 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 vsprintf	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 Speicher manipulation	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

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	Mutimedia-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

2 Das erste Programm

Ziel von Kapitel 2 ist es, sich einen ersten Überblick über den Aufbau eines Programms zu verschaffen. Außerdem soll das Programm ausführbar gemacht, also übersetzt werden. Voraussetzung für dieses Kapitel ist, dass bereits ein Compiler installiert wurde.

2.1 Der beste Lernerfolg

Vielleicht haben Sie in diesem Buch schon ein wenig geblättert und sich die hinteren Kapitel angesehen. Wenn Sie gerade dabei sind, C zu lernen oder auch C Ihre erste Programmiersprache ist, dürfte vieles auf den hinteren Seiten ein wenig kryptisch auf Sie wirken. Als Vergleich könnte ich jetzt eine beliebige Fremdsprache nehmen, die Sie gerade lernen wollen. Wenn Sie dabei die hinteren Kapitel ansehen, wird es Ihnen genauso gehen wie mit diesem Buch, Sie werden mit den Fremdwörtern auch noch nichts anfangen können. Den besten Lernerfolg erzielen Sie also, indem Sie das Buch von vorn nach hinten durcharbeiten.

Wer C wirklich lernen will, sollte die Programme auch in der Praxis ausführen. Ideal wäre es natürlich, wenn Sie die Beispiele abtippen und ein wenig damit experimentieren, eventuell sogar mit Absicht Fehler einbauen und sich die Meldungen des Compilers ansehen, mit dem Ziel, möglichst viel daraus zu lernen.

Erhoffen Sie sich, dieses Buch in wenigen Tagen durchgearbeitet zu haben und dann ein Profi zu sein, werden Sie schnell enttäuscht sein. Programmieren zu lernen, dauert ein wenig länger. Wenn Sie aber Geduld haben und immer am Ball bleiben, ist der Grundstein einer Programmiererkarriere gelegt. Das »Programmieren lernen« beruht wie vieles im Leben auf dem Prinzip »Versuch und Irrtum«.

2.2 »Hallo Welt« in C

Obligatorisch ist es, mit dem berühmten »Hallo Welt«-Programm in C zu beginnen. Ziel dieses Programms ist es, die Textfolge »Hallo Welt« auf dem Bildschirm auszugeben. Tippen Sie im Texteditor (bzw. der Entwicklungsumgebung) folgendes Programm ein:

```
/* hallo.c */  
#include <stdio.h>
```

```
int main (void) {
    printf("Hallo Welt\n");
    return 0;
}
```

Zunächst wird das Programm in einem Verzeichnis, wo auch immer das Programm zur Ausführung gebracht werden soll, mit der Endung – ebenso Extension genannt – »*.c« abgespeichert (wobei »*« für den beliebigen Namen des Programms steht).

Jetzt müssen Sie das Programm übersetzen (kompilieren). Dieser Vorgang kann von Compiler zu Compiler verschieden ablaufen. Bei einer Entwicklungsumgebung muss dabei häufig nur der Button »Compile« oder »Kompilieren« angeklickt werden. In einer Konsole wird das Programm mit der Kommandozeile übersetzt.

Tipp Wird das Programm in einer Entwicklungsumgebung unter Windows übersetzt, sollten Sie vor der Zeile `return 0;` immer die Funktion `getchar();` schreiben, damit das Programm kurz anhält und sich nicht gleich wieder beendet (genauer: die MS-DOS-Konsole sich nicht gleich wieder beendet). Damit wartet das Programm bis die ENTER-Taste betätigt wurde, bevor es sich beendet.

Hinweis Anleitungen, wie Sie ein Listing mit einem bestimmten Compiler übersetzen können, habe ich Ihnen unter der Webadresse <http://www.pro-nix.de/> bereitgestellt.

Als Ergebnis findet sich im Verzeichnis ein ausführbares Programm namens »hallo« bzw. »hallo.exe« (sollten Sie das Programm »hallo.c« genannt haben). Diese Datei kann jetzt wie jede andere ausführbare Datei in der Kommandozeile gestartet werden.

Bei Entwicklungsumgebungen dürfte meistens ein Klick auf »Ausführen« bzw. »Run« reichen. Wenn die Textfolge »Hallo Welt« auf dem Bildschirm ausgegeben wird, ist es geschafft. Sie haben das erste Programm geschrieben und erfolgreich übersetzt!

Hinweis Sollten Sie das Programm unter Linux nicht mit dem Programmnamen starten können, schreiben Sie einfach ein `./` vor den Programmaufruf. Beispielsweise: `./programmname`

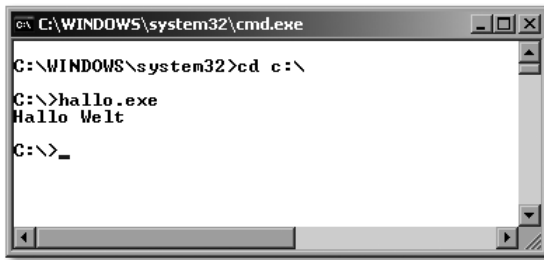


Abbildung 2.1 »Hallo Welt« in einem Konsolenfenster unter MS-DOS

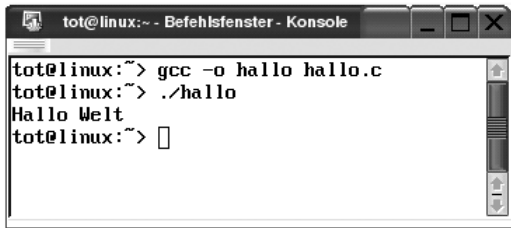


Abbildung 2.2 Programm »Hallo Welt« in einer Konsole unter Linux

2.3 Analyse des Programms

Jetzt soll das erste Programm in seine einzelnen Bestandteile zerlegt und Schritt für Schritt durchgegangen werden.

```
#include <stdio.h>
```

`include` ist kein direkter Bestandteil der Sprache C, sondern ein Befehl des Präprozessors. Der Präprozessor ist ein Teil des Compilers, der nicht das Programm übersetzt, sondern kontrollierend nicht bleibende Änderungen im Programmtext vornimmt. Diese Änderungen sind jedoch nur temporär.

Präprozessorbefehle erkennen Sie am vorangestellten `#`-Zeichen. Kompilieren Sie das Programm noch einmal ohne `#include <stdio.h>`, dann sollte eine Fehlermeldung folgen, wie z.B. die folgende:

```
Error. function 'printf' should have a prototype.
```

`printf()` kommt doch im Programm vor? Richtig. `printf()` ist eine (Standard-)Funktion, die in `#include <stdio.h>` deklariert ist. `include`-Dateien nennt man auch Headerdateien. Suchen Sie das Verzeichnis `INCLUDE` auf Ihrem System (unter Linux ist das typischerweise `/usr/include` oder `/usr/bin/include` und bei MS Windows z.B. `C:\Name_des_Compilers\include`), und Sie werden noch viele andere Headerdateien darin entdecken, die später noch Verwendung finden.

Die Abkürzung `stdio` steht für **Standard-Input/Output**, also Standard-Ein- und Ausgabe. Es wird noch öfters auf die Headerdateien eingegangen, die ohnehin

in jedem Programm benötigt werden. Später werden Sie auch eigene Headerdateien entwerfen und im Programm einbinden.

Weiter zur Programmausführung:

```
int main(void)
```

Hier beginnt das Hauptprogramm. Eine `main`-Funktion wird immer benötigt, damit der Compiler weiß, wo er beginnen muss, das Programm zu übersetzen. Auch wenn später mehrere Module (Funktionen), also mehrere Quellcode-Dateien, kompiliert werden (zu einer ausführbaren Datei binden), benötigen Sie immer eine `main`-Funktion. `main` heißt auf Deutsch so viel wie Hauptfunktion. Das `void` in der `main()`-Funktion steht für einen »leeren« Datentyp (mehr dazu später). `void` könnten Sie hier auch ganz weglassen – allerdings soll hier nicht unerwähnt bleiben, dass `()` und `(void)` in C++ nicht dasselbe sind. Sofern Sie also Ihre Programme erweitern wollen, sollten Sie es gleich mit angeben.

`int` steht für eine Ganzzahl. Im Fall einer Funktion bedeutet dies, dass diese einen Rückgabewert hat, vom Typ `int`. In diesem Programm bekommt die `main`-Funktion den Rückgabewert `0` durch den Aufruf:

```
return 0;
```

Was bedeutet: Das Programm wurde ordnungsgemäß beendet. Es wird also mit `return` hier der Funktion `main` der Wert `0` zurückgegeben. Genaueres dazu (zur `main()`-Funktion und deren Rückgabewerte) in einem späteren Kapitel. Weiter mit:

```
{  
    printf(".....");  
}
```

Zwischen den geschweiften Klammern steht der Anweisungsblock. Das heißt, in diesem Block befinden sich alle Anweisungen, die die Funktion `int main()` auszuführen hat. Natürlich können innerhalb eines Anweisungsblocks weitere Anweisungsblöcke verwendet werden. Das hört sich komplizierter an als es ist. Darauf wird später noch eingegangen.

Merke Geschweifte Klammern fassen Anweisungen zu einem Block zusammen.

Und was geschieht in diesem Anweisungsblock?


```
printf("Hallo Welt\n");
```

`printf()` ist eine Funktion, die in der Headerdatei `stdio.h` deklariert ist, wie bereits erwähnt wurde. Deswegen kann der Compiler, wenn Sie diese Headerdatei nicht im Programm angegeben haben, nichts mit `printf()` anfangen. Mit der Funktion `printf()` kann eine beliebige Stringkonstante formatiert auf dem Bildschirm ausgegeben werden. Die Stringkonstante, in diesem Fall »Hallo Welt«, die ausgegeben wird, steht immer zwischen zwei Hochkommata ("Stringkonstante"). Nicht erlaubt ist es, eine Stringkonstante über das Zeilenende fortzusetzen, wie etwa im folgenden Beispiel:

```
printf("Dies ist in C
nicht erlaubt");
```

Es gibt aber eine Ausnahme dieser Regel: indem Sie ein `\` (Backslash) setzen. Hierzu ein Beispiel:

```
printf("Hier ist die Ausnahme der Regel \
dies hier ist erlaubt, dank Backslash");
```

Sie sollten aber dabei beachten, dass alle Leerzeichen nach dem Backslash in der nächsten Zeile ebenfalls bei der Ausgabe berücksichtigt werden. Das Zeichen `'\n'` in der Funktion von `printf()` ist ein Steuerzeichen und bedeutet *newline* und erzeugt auf dem Bildschirm einen Zeilenvorschub, wie er mit der Tastatur durch die Taste  ausgelöst wird.

Jetzt zum Semikolon (;): Es wird hauptsächlich dazu verwendet, das Ende einer Anweisung anzuzeigen. Der Compiler weiß dann, hier ist das Ende der Anweisung von `printf()` und fährt nach der Abarbeitung der Anweisung mit der nächsten Zeile bzw. Anweisung fort. Natürlich hat das Semikolon keine Wirkung, wenn es in einer Stringkonstante verwendet wird:

```
printf("Hallo; Welt\n");
```

Ausgabe:

```
Hallo; Welt
```

Merke Anweisungen, denen kein Anweisungsblock folgt, werden mit einem Semikolon abgeschlossen.

Index

- (Operator) 98
-- (Operator) 102
! (Operator) 127
!= (Operator) 120
#define-Anweisung 203
#elif-Anweisung 213
#else-Anweisung 213
#endif-Anweisung 213
#error-Anweisung 221
#ifdef-Anweisung 212
#include-Anweisung 199
#line-Anweisung 218
#pragma pack 430
#pragma-Anweisung 222
#undef-Anweisung 211, 220
% (Operator) 98
%= (Operator) 101
& (Operator) 52, 104
&& (Operator) 129
(type)Operator 115
* (Operator) 98
*= (Operator) 101
+ (Operator) 98
++ (Operator) 102
+= (Operator) 101
. (Operator) 386
/ (Operator) 98
/= (Operator) 101
< (Operator) 120
<< (Operator) 108
<= (Operator) 120
-= (Operator) 101
== (Operator) 120
-> (Operator) 393, 673
> (Operator) 120
>= (Operator) 120
>> (Operator) 109
?: (Operator) 132
^ (Operator) 107
__attrib__ 429
__cplusplus Makro 217
__DATE__ Makro 217
__FILE__ 621
__FILE__ Makro 217
__func__ (C99) 218, 665
__LINE__ 621
__LINE__ Makro 217
__STD_HOSTED__ (C 99) 219, 665
__STD_VERSION__ (C 99) 219, 665

__STDC__ Makro 217
__TIME__ Makro 217
_exit() 628
_IOFBF 515
_IOLBF 515
_IONBF 515
| (Operator) 106
|| (Operator) 129
~ (Operator) 108
o_EXCL 520

A

abnormale Programmbeendigung 630
abort() 621, 630
abs() 639
Absolutwert 627, 639
accept() 1051
access() 580
acos() 626
Adressoperator 52, 284
AF_BLUETOOTH 1038
AF_INET 1038
AF_INET6 1038
AF_IRDA 1038
AF_UNIX 1038
Algorithmen 747
alloca() 375
ALTER TABLE 964
ANSI 29
ANSI C
 C99 662
ANSI C Standard
 C89 Standard 28
 C99 Standard 28
Anweisungsblock 40, 117
Apache Webserver 891
 htdocs-Verzeichnis 896
 installieren 892
 Konfiguration 897
 starten und stoppen 896
Arcuscosinus 626
Arcussinus 626
Arcustangens 626
Array
 Anzahl der Elemente ermitteln 233
 Bereichsüberschreitung 225
 Deklarieren 223
 dynamisch 367, 376
 Funktionsübergabe 234
 Funktionsübergabe als Kopie 235

- Gültigkeitsbereich 229
- Indizierungsoperator 225
- Initialisierung und Zugriff 224
 - mehrdimensional 242, 376
 - Rückgabe aus Funktion 236
 - Strukturen 397
 - Vergleichen 231
 - von Tastatur einlesen 241
- Array und Zeiger 305, 334, 379
- ASCII-Zeichensatz 82
- asctime() 613
- asin() 626
- assert() 621
- atan() 626
- atan2() 626
- atexit() 629
- atof() 631
- atoi() 631
- atol() 631
- Attribute einer Datei 563
- Aufzählungstyp
 - enum 421
- ausgeglichener Binärbaum 802
- auto 169
- AVL-Baum 803

B

- Backtracking 833
- Bäume 780
- B-Baum 804
- bedingte Compilierung 211
- Bedingung
 - else 123
 - else if 122
 - if 117
- Bedingungsoperator 132
- Bezeichner 67
- Big Endian 503
- Bildschirm löschen 212
- Binäre Bäume 780
 - Ast, Kante 781
 - ausgleichen 802
 - AVL-Baum 803
 - B-Baum 804
 - Blatt 781
 - Elternzeiger 801
 - Entartet 802
 - Knoten 781
 - Teilbaum 781
 - Threads 802
 - Tiefe 781
 - Wurzel 781

- binäre Suche 777
- bind() 1048
- Bitfelder 431
 - anonym 437
- Bitmanipulationen
 - Bits abfragen 109
 - Bits löschen 109
 - Bits setzen 109
- Bitweise Operatoren 103
- Blockweise (binär)
 - lesen 496
 - schreiben 496
- Boyer-Moore-Algorithmus 826
- break 155
- bsearch() 642
- Bubble Sort 752
- Buffer Overflow 858
 - anfällige Funktionen 868
 - fgets() 267
 - Gegenmaßnahmen 868
 - gets() 482, 869
 - printf() 62, 870
 - Programme und Tools 874
 - scanf() 57, 869
 - sprintf() 870
 - strcat() 269, 869
 - strcpy() 272, 859, 869
 - Wrapper-Funktion 872
- BUFSIZ 510
- Byte stream 441

C

- C89 Standard 29
- C99 662
- C99 Standard 29
- call by reference 234, 298
- call by value 174
- calloc() 371
- C-API
 - MySQL 971
- Casting 115
- ceil() 627
- CGI 883
 - Anwendung ausführen 899
 - Anwendung erstellen 899
 - Ausgabe 917
 - CGI-Schnittstelle 883, 911
 - CONTENT_LENGTH 928
 - Eingabe 927
 - Eingabe dekodieren 931
 - Eingabe parsen 931
 - Filehandle 911

- Formular-Tags 920
- Gästebuch 936
- getenv() 914
- GET-Methode 905, 928
- HTTP Request 905
- HTTP-Protokoll 901
- Methode auswerten 928
- mit der MySQL C-API 1003
- POST-Methode 906, 928
- QUERY_STRING 928
- Query-String 904
- Request Header 907
- Server Response 908
- Statuscode 909
- Umgebungsvariablen 912
- Zugriffsrechte 899
- char 77
- CHAR_BIT 85
- CHAR_MAX 85
- CHAR_MIN 85
- chdir() 585
- clock() 615, 618
- clock_t 607
- CLOCKS_PER_SEC 608, 615
- close() 545, 1047
- closedir() 592
- closesocket() 1047
- clearerr() 472
- clrscr() 212
- connect() 1039
- const 170
- continue 154
- cos() 626
- cosh() 626
- Cosinus 626
- creat() 546
- CREATE DATABASE 960
- CREATE TABLE 962
- Cross-Plattform-Entwicklung 1054
 - Abstraktion Layer 1062
- CSV-Datei 258, 463
- ctime() 613

D

- Datei
 - Änderungszeit 571
 - Attribute 563
 - Binärmodus 447
 - Eigentümer 576
 - EOF-Flag 471
 - erstellen (Low Level) 539, 546
 - Fehler-Flag 472

- Größe 570
- Gruppeneigentümer 576
- Inode-Änderung 571
- Lesen (blockweise, binär) 496
- Lesen (formatiert) 462
- Lesen (Low Level) 551
- Lesen einer Zeile 481
- Lesen eines Zeichen 449, 453
- löschen 507
- neu anlegen 446
- öffnen 443, 446, 448
- öffnen (Low Level) 538
- positionieren 476, 480
- positionieren (Low Level) 558
- Pufferung 475, 510
- schließen 459
- schließen (Low Level) 545
- schreiben (blockweise, binär) 496
- schreiben (formatiert) 462
- schreiben (Low Level) 547
- schreiben einer Zeile 485
- schreiben eines Zeichen 449, 453
- temporäre 516
- Textmodus 447
- umbenennen 509
- Zugriffsrechte 540, 567, 580
- Zugriffszeit 571
- Dateiarten 564
 - S_IFCHR 564
 - S_IFDIR 564
 - S_IFREG 564
 - S_ISBLK 564
 - S_ISFIFO 564
 - S_ISLNK 564
 - S_ISSOCK 564
 - st_mode 564
- Datenbanksystem 951
 - relationales Modell 954
- Datentyp
 - char 77
 - double 70
 - float 70
 - int 63
 - long 68
 - long double 73
 - long long 662
 - short 69
- Datum 607
 - tm-Struktur 607
- DBL_DIG 87
- DBL_EPSILON 87
- DBL_MANT_DIG 87

DBL_MAX 87
 DBL_MAX_10_EXP 87
 DBL_MAX_EXP 87
 DBL_MIN 88
 DBL_MIN_10_EXP 87
 DBL_MIN_EXP 87
 DBMS 952
 Drei-Schichten-Modell 952
 Deadlock 738
 default 137
 Deklaration und Definition 65
 DELETE FROM 968
 Dennis Ritchie 28
 Dereferenzierungsoperator 287
 Differenzen C und C++
 Deklaration von Variablen 68
 sizeof-Operator 113
 difftime() 614
 dirent-Struktur 592
 div() 640
 div_t 640
 do while-Schleife 142
 Domainname 1030
 doppelt verkettete Listen 698
 double im Detail 73
 DROP DATABASE 962
 DROP TABLE 964
 dynamische Datenstrukturen 671
 dynamische Speicherverwaltung 351
 dynamisches Array 367
 dynamisches Array zweidimensional
 376

E

EDOM 521
 Ein-/Ausgabe-Funktionen
 höhere Ebene 442
 niedrigere Ebene 537
 einfach verkettete Listen 671
 elementare Datentypen 63
 Elementkennzeichnungsoperator 393
 else if-Bedingung 122
 else-Bedingung 123
 Endlosschleife 140
 entarteter Binärbaum 802
 Entwicklungsumgebung 34
 enum 421
 EOF 450, 471
 ERANGE 521
 errno 521
 exit() 628
 EXIT_FAILURE 181

EXIT_SUCCESS 181
 exp() 627
 EXPLAIN 963
 extern 169, 172

F

fabs() 627
 Fakultät 191
 fclose() 459
 FD_CLR() 1081
 FD_ISSET() 1081
 FD_SET() 1081
 FD_ZERO() 1081
 fdopen() 560
 Fehlerausgabe 521
 perror() 522
 strerror() 523
 feof() 471
 ferror() 472
 fflush() 55, 474
 fgetc() 453
 fgetpos() 480
 fgets() 267, 481
 Fibonacci-Zahlen 192
 File-Deskriptor 537
 fileno() 559
 FILE-Struktur 444
 float im Detail 72
 floor() 627
 FLT_DIG 87
 FLT_EPSILON 87
 FLT_MANT_DIG 87
 FLT_MAX 87
 FLT_MAX_10_EXP 87
 FLT_MAX_EXP 87
 FLT_MIN 87
 FLT_MIN_10_EXP 87
 FLT_MIN_EXP 87
 FLT_RADIX 87
 fmod() 627
 fopen() 443
 FOPEN_MAX 448, 459
 formatierte Ausgabe
 fprintf() 462
 printf() 61
 sprintf() 525
 formatierte Ein-/Ausgabe 441
 Umwandlungsvorgaben 90
 formatierte Eingabe
 fscanf() 462
 scanf() 51
 sscanf() 526

- Formatierungsanweisung 90
- Formatstring von printf() 61
- Formular
 - Auswahlliste 925
 - Beginn und Ende 921
 - Checkbox 923
 - Eingabefeld 921
 - Passwort-Feld 922
 - Radiobutton 923
 - Reset-Button 924
 - Submit-Button 924
 - Textfeld 924
- for-Schleife 146
- fpos_t 480
- fprintf() 462
- fputc() 456
- fputs() 485
- fread() 497
- free() 360, 375
 - Makro 362
- Freispeicherverwaltung 363
 - Best-Fit-Verfahren 366
 - Buddy-Verfahren 366
 - First-Fit-Verfahren 365
 - Next-Fit-Verfahren 365
 - Quick-Fit-Verfahren 366
 - Worst-Fit-Verfahren 366
- freopen() 506
- frexp() 627
- fscanf() 462
- fseek() 476
- fsetpos() 480
- ftell() 479
- Funktion
 - Argument 176
 - Array als Rückgabewert 236
 - call by reference 298
 - call by value 174
 - Definition von Funktionen 159
 - Funktionsaufruf 160
 - Funktionsdeklaration 162
 - main() 179
 - Parameter 175
 - Rekursion 185
 - Rückgabewert 178
 - Speicherklassen 171–172
 - Struktur (call by reference) 392
 - Struktur (call by value) 391
 - Struktur als Rückgabewert 393
 - Übergabe eines Arrays 234
 - Vorwärtsdeklaration 162
 - Zeiger als Rückgabewert 302

- Funktionen
 - mehrdimensionale Arrays 256
- fwrite() 498
- G**
- Ganzzahlen
 - char 77
 - int 63
 - long 68
 - short 69
- Ganzzahlkonstanten 88
- Geltungsbereich von Variablen 171
- Genauigkeit von Gleitpunktzahlen 74
- getc() 453
- getch() 451
- getchar() 449
- getcwd() 589
- getenv() 914, 983
- gethostbyname() 1042
- gets() 481
- getservbyname() 1041
- Gleitpunktkonstanten 89
- Gleitpunktzahlen
 - double 70
 - float 70
 - Genauigkeit 74
 - long double 73
 - numerische Gleitpunktprobleme 76
- Gleitpunktzahlen-Umgebung 669
- gmtime() 610
- goto 155
- Größe
 - Datei (st_size) 570
- größter gemeinsamer Teiler 193
- GUI-Programmierung 1091

- H**
- Hashing 804
 - direkte Adressierung 814
 - Hash-Funktion 809
 - Hashfunktion 804
 - Hash-Tabelle 805
 - von Strings 810
- Headerdatei
 - <assert.h>-Headerdatei 621
 - <ctype.h>-Headerdatei 623
 - <errno.h>-Headerdatei 521
 - <fcntl.h>-Headerdatei 669
 - <float.h>-Headerdatei 85
 - <inttypes.h>-Headerdatei 669
 - <iso646.h>-Headerdatei (C 99) 663
 - <limits.h>-Headerdatei 85
 - <locale.h>-Headerdatei 645

<math.h>-Headerdatei 626, 666
<setjmp.h>-Headerdatei 648
<signal.h>-Headerdatei 653
<stdarg.h>-Headerdatei 597
<stdint.h>-Headerdatei (C 99) 662
<stdio.h>-Headerdatei 442
<stdlib.h>-Headerdatei 628
<string.h>-Headerdatei 268, 658
<sys/stat.h>-Headerdatei 541, 563
<tgmath.h>-Headerdatei 669
<time.h>-Headerdatei 607
<wchar.h>-Headerdatei 668
<wctype.h>-Headerdatei 668
Heap 352, 867, 876
Hostname 1030
HTML 885
 Formular-Tags 920
 Tag 920
htonl() 1041
htons() 1041
HTTP-Protokoll 888, 901
HUGE_VAL 628

I

IEEE-Standard 754 72
if-Bedingung 117
INADDR_ANY 1049
Indizierungsoperator 225, 397
inline 211, 664
INSERT INTO 965
Insertion Sort 750
int 63
INT_MAX 86
INT_MIN 86
IP-Nummer 1028
 dynamisch 1028
 statisch 1028
IP-Protokoll 1031
IPv4 1028
IPv6 1028
isalnum() 623
isalpha() 623
isascii() 623
isblank() 623
iscntrl() 623
isdigit() 623
isgraph() 623
islower() 623
isprint() 623
ispunct() 623
isspace() 623
isupper() 623

isxdigit() 623
itoa() 637

J

jmp_buf 650

K

Karp-Rabin-Algorithmus 826
KMP-Algorithmus 819
Kommandozeile 339
 Argumente 339
 Optionen 345
Kommandozeilenprozessor 644
Kommentar 47
Komplexe Gleitpunktzahlen 663
Konstante 88
 Betriebssystem 215
 Compiler 214

L

labs() 639
Laufzeitmessung 618
lconv-Struktur 647
LDBL_DIG 87
LDBL_EPSILON 87
LDBL_MANT_DIG 87
LDBL_MAX 87
LDBL_MAX_10_EXP 87
LDBL_MAX_EXP 87
LDBL_MIN 88
LDBL_MIN_10_EXP 87
LDBL_MIN_EXP 87
ldiv() 640
ldiv_t 640
LIFO-Prinzip 715
lineare Suche 775
Linksassoziativität 97
listen() 1050
Little-Endian 503
load_defaults() 981
localeconv() 647
localtime() 610
log() 627
log10() 627
Logarithmus 627
logische Operatoren 127, 129
long 68
long double 70, 73
long long 662
long long (C99) 69
LONG_MAX 86
LONG_MIN 86
longjmp() 650

Loopback-Interface 1036
lseek() 558

M

main() 339
main()-Funktion 179
MAKEWORD() 1035
Makro 208, 665
malloc() 352
 Typen-Casting 353
Maschinencode 863
mathematische Funktionen 626, 666
MB_LEN_MAX 85
memchr() 658
memcmp() 232, 659
memcpy() 660
memmove() 660
Memory Leak 351, 376, 876
 Bibliotheken und Tools 880
 Gegenmaßnahmen 877
memset() 228, 661
mkdir() 583
mktime() 611
modf() 627
Modulo-Operator 99
MSQL_UNIX_SOCKET 983
MySQL 951
 Befehle 970
 Bibliothek 972
 C-API 971, 975
 Datentypen 958
 eigene Clientprogramme 971
 Headerdateien 972
 Installation 956
 libmysql.dll 972–973
 libmysql.lib 972
 mysql-Client starten 957
 Server starten 956
MySQL C-API
 Anfrage an den Server 983
 CGI-Anwendung 1003
 Daten ausgeben 1000
 Daten hinzufügen 989, 1004
 Daten löschen 991
 Daten suchen 998, 1013
 Daten verändern 993
 Daten vom Server holen 996
 Datenbank auswählen 989
 Datentypen 1025
 Fehlerüberprüfung 978
 Feldwert ermitteln 999
 Funktionsübersicht 1022

Kommandozeilen-Argumente 980
MYSQL_FIELD-Struktur 999
MYSQL_RES-Struktur 996
MYSQL-Handle 976
 Serverantwort auswerten 993
 Verbindung herstellen 975
 Verbindung schließen 979
mysql_affected_rows() 993, 1022
mysql_change_user() 1022
mysql_character_set_name() 1022
mysql_close() 979, 1022
mysql_connect() 1022
mysql_create_db() 1022
mysql_data_seek() 1022
mysql_debug() 1023
mysql_drop_db() 1023
mysql_dump_debug_info() 1023
mysql_eof() 1023
mysql_errno() 978, 1023
mysql_error() 979, 1023
mysql_escape_string() 1023
mysql_fetch_field() 999, 1023
mysql_fetch_field_direct() 1023
mysql_fetch_fields() 1023
mysql_fetch_lengths() 1023
mysql_fetch_row() 997, 1023
mysql_field_count() 1023
mysql_field_seek() 1003, 1023
mysql_field_tell() 1003, 1023
mysql_free_result() 997, 1023
mysql_get_client_info() 1023
mysql_get_host_info() 1023
mysql_get_proto_info() 1023
mysql_get_server_info() 1024
mysql_info() 1024
mysql_init() 976, 1024
mysql_insert_id() 1024
mysql_kill() 1024
mysql_list_dbs() 1024
mysql_list_fields() 1024
mysql_list_processes() 1024
mysql_list_tables() 1024
mysql_num_fields() 997, 1024
mysql_num_rows() 997, 1024
mysql_options() 1024
mysql_ping() 1024
mysql_query() 984, 1024
mysql_real_connect() 976, 1024
mysql_real_escape_string() 1023
mysql_real_query() 984, 1024
mysql_reload() 1024
mysql_row_seek() 1024

- mysql_row_tell() 1024
- mysql_select_db() 989, 1024
- mysql_shutdown() 1024
- mysql_stat() 1025
- mysql_store_result() 996, 1025
- MYSQL_TCP_PORT 983
- mysql_thread_id() 1025
- mysql_thread_safe() 1025
- mysql_use_result() 1025
- MySQL-Clientprogramme
 - mit Borland-Compiler 974
 - mit gcc 973
 - mit Visual C++ 973
 - Probleme beim Erstellen 975

N

- Nameserver 1030
- NDEBUG 622
- Netzwerkprogrammierung 1027
 - Client-/Server-Prinzip 1036
 - Linux 1033
 - Windows 1033
- Netzwerktechnik 1027
 - Domainname 1030
 - Hostname 1030
 - IP-Nummer 1028
 - IP-Protokoll 1031
 - IPv4 1028
 - IPv6 1028
 - Loopback-Interface 1036
 - Nameserver 1030
 - OSI-Schichtmodell 902
 - Portnummer 1029
 - Sockets 1032
 - TCP 1031
 - UDP 1031
- Notationsstil 157
- ntohl() 1041
- ntohs() 1041
- NULL 291, 353, 356, 608

O

- O_APPEND 539
- O_BINARY 540
- O_CREAT 539
- O_EXCL 540
- O_NOCTTY 540
- O_NONBLOCK 540
- O_RDONLY 539
- O_RDWR 539
- O_SYNC 540
- O_TEXT 540
- O_TRUNC 540

- O_WRONLY 539
- objdump 862
- ODBC-Schnittstelle 955
- offsetof-Makro 438
- Opcodes 863
- open() 538
- OPEN_MAX 545
- opendir() 591
- Operatoren
 - arithmetische 98
 - bitweise 103
 - Dekrement 102
 - Inkrement 102
 - logische 127
 - Zuweisungs- 101

P

- Paging 364
- Parameter
 - von main() 339
- Pattern Matching 826
- perror() 522
- Portabel
 - getch() 451
 - Verzeichnis ausgeben 594
- Portnummer 1029
- pow() 627
- Präprozessor 199
- Präprozessor-Direktiven
 - define 203
 - elif 211
 - else 211
 - endif 211
 - error 221
 - ifdef 211
 - include 199
 - line 218
 - pragma 222
 - undef 220
- printf() 61
- Profiler 618
 - gprof 619
- Programmbeendigung 628
 - _exit() 628
 - abort() 630
 - atexit() 629
 - exit() 628
- Prozess 181
- ptrdiff_t 297
- Pufferüberlauf 858
- Pufferung 475
 - einstellen 510

Punktoperator 386
putc() 456
putchar() 449
puts() 485

Q

qsort() 641, 764
Quadratwurzel 627
Queues 737
 get() 737
 put() 737
Quicksort 758

R

raise() 657
rand() 638
RAND_MAX 638
read() 551
readdir() 591
Read-only-Zeiger 314
realloc() 371
Rechtsassoziativität 97
recv() 1045
recvfrom() 1046
register 169
reguläre Ausdrücke 826
Rekursion 185
relationale Beziehung 955
remove() 507
rename() 509
return 178
rewind() 480
rewinddir() 592
rmdir() 586
Rückgabewert beim Beenden 181
Rückgabewert von Funktionen 178
Rückgabewert von main() 179
Rücksprungadresse manipulieren 861

S

S_IEXEC 568
S_IREAD 541, 568
S_IRGRP 541, 567
S_IROTH 541, 567
S_IRUSR 541, 567
S_IRWXG 541
S_IRWXO 542
S_IRWXU 541
S_ISGID 541
S_ISUID 541
S_ISVTX 541
S_IWGRP 541, 567
S_IWOTH 542, 567

S_IWRITE 541, 567
S_IWUSR 541, 567
S_IXGRP 541, 567
S_IXOTH 542, 567
S_IXUSR 541, 567
scanf() 51
SCHAR_MAX 85
SCHAR_MIN 85
Schleife
 do while 142
 Endlosschleife 140
 for 146
 while 138
Schlüsselwörter in C 67
SEEK_CUR 476, 558
SEEK_END 476, 558
SEEK_SET 476, 558
seekdir() 593
SELECT ... FROM 966
select() 1079
 FD_CLR() 1081
 FD_ISSET() 1081
 FD_SET() 1081
 FD_ZERO() 1081
Selektion Sort 748
send() 1045
sendto() 1046
setbuf() 510
setjmp() 650
setlocale() 646
setvbuf() 514
Shellsort 754
 Distanzfolgen 755
short 69
SHOW DATABASE 961
SHOW TABLES 963
SHRT_MAX 85
SHRT_MIN 85
sicheres Programmieren 857
 Buffer Overflow 858
 Memory Leak 876
Sicherheitsprobleme vermeiden 881
Signal
 SIG_DFL 654
 SIG_ERR 657
 SIG_IGN 654
 SIGABRT 653, 657
 SIGFPE 653
 SIGILL 653
 SIGINT 653–654
 SIGSEGV 653
 SIGTERM 653

- signal() 653
- Signalkonzept 653
- signed 83
- sin() 626
- sinh() 627
- Sinus 626
- size_t 607
- sizeof-Operator 111, 233, 386
- socket() 1037
- Socketprogrammierung 1033
 - accept() 1051
 - Adresse festlegen (binden) 1048
 - bind() 1048
 - Client-/Server-Anwendung 1076
 - Client-/Server-Prinzip 1036
 - Client-Anwendung 1037
 - close() 1047
 - closesocket() 1047
 - connect() 1039
 - Daten empfangen 1044
 - Daten senden 1044
 - Datenformate 1086
 - gethostbyname() 1042
 - hostent 1043
 - htonl() 1041
 - htons() 1041
 - IPv4 nach IPv6 ändern 1088
 - Linux 1033
 - listen() 1050
 - Mehrere Clients gleichzeitig 1078
 - ntohl() 1041
 - ntohs() 1041
 - Portabilität 1088
 - Puffer 1087
 - recv() 1045
 - recvfrom() 1046
 - RFC-Dokumente 1090
 - select() 1079
 - send() 1045
 - sendto() 1046
 - servent 1042
 - Server-Anwendung 1048
 - Serverhauptschleife 1051
 - Sicherheit 1090
 - sockaddr_in 1040
 - Socket anlegen 1037
 - Socket freigeben 1047
 - socket() 1037
 - TCP-Echo-Server 1054
 - Verbindung annehmen 1051
 - Verbindung herstellen 1039
 - Warteschlange einrichten 1050
 - Windows 1033
 - Sockets 1032
 - Sortieralgorithmus 748
 - analysieren 766
 - Bubble Sort 752
 - Insertion Sort 750
 - qsort() 764
 - Quicksort 758
 - Selektion Sort 748
 - Shellsort 754
 - Sortieren 748
 - qsort() 641
 - Speicher allozieren
 - alloca() 375
 - calloc() 371
 - malloc() 352
 - realloc() 371
 - Speicher freigeben
 - free() 360, 375
 - Speicherhierarchie 364
 - Speicherleck 876
 - Speicherverwaltung 859
 - sprintf() 525
 - Sprung
 - longjmp() 650
 - setjmp() 650
 - Sprunganweisung
 - break 155
 - continue 154
 - SQL 951
 - Befehle 960
 - Crashkurs 958
 - Daten abfragen 966
 - Daten eingeben 965
 - Daten löschen 968
 - Daten verändern 968
 - Datenbank anlegen 960
 - Datenbank ausgeben 961
 - Datenbank auswählen 962
 - Datenbank löschen 961
 - Datensatz 955
 - Datentypen 958
 - DDL-Anweisungen 958
 - DML-Anweisungen 958
 - SQL-Kommandos 970
 - Tabelle anlegen 962
 - Tabelle anzeigen 963
 - Tabelle löschen 964
 - Tabelle verändern 964
 - Tabellen-Details 963
 - sqrt() 627
 - srand() 638

- sscanf() 526
- Stack 186, 650, 715, 860
 - debuggen 736
 - pop() 715
 - push() 715
- Stack-Frame 860
- Stackrahmen 176
- Standard-Headerdateien 202
- Standard-Streams 467
 - umleiten 468
- Startup-Code 180
- stat() 563
- static 170, 172
- stat-Struktur 563
 - st_atime 571
 - st_ctime 571
 - st_dev 577
 - st_gid 576
 - st_ino 577
 - st_mode 564
 - st_mtime 571
 - st_nlink 577
 - st_rdev 577
 - st_size 570
 - st_uid 576
- stderr 467
- STDERR_FILENO 470
- stdin 467
- STDIN_FILENO 470
- stdout 467
- STDOUT_FILENO 470
- Steuerzeichen 43
- strcat() 269
- strchr() 270
- strcmp() 270
- strcpy() 271
- strcspn() 272
- Stream 441–442, 444
- strerror() 523
- strftime() 616
- String
 - einlesen 266
 - lesen (formatiert) 526
 - reguläre Ausdrücke 826
 - schreiben (formatiert) 525
 - Stringmatching 815
 - umwandeln in numerischen Wert 631
- Stringende-Zeichen 260
- Stringkonstante 89, 260
- Stringmatching-Algorithmus 815
 - Boyer-Moore-Algorithmus 826
 - Brute Force 816
 - Karp-Rabin-Algorithmus 826
 - Knuth/Morris/Pratt-Algorithmus 819
 - naiver Algorithmus 816
 - reguläre Ausdrücke 826
- Strings 259
- Strings aneinander hängen
 - strcat() 269
 - strncat() 273
- Stringtabelle 315–316
- strlen() 272
- strncmp() 274
- strncpy() 275
- strpbrk() 276
- strrchr() 276
- strspn() 277
- strstr() 277
- strtod() 633
- strtok() 278
- strtol() 634
- strtoul() 634
- struct 385
- Struktur
 - an Funktion übergeben 391
 - Array 397
 - Attribute verändern 428
 - Bitfelder 431
 - Deklarieren 385
 - in Struktur 404
 - Initialisierung und Zugriff 386
 - Lebensdauer 385
 - offsetof-Makro 438
 - Rückgabewert von Funktion 393
 - Union 416
 - vergleichen 395
- Strukturelemente 386
- Strukturen 385
- Suchalgorithmus 774, 815
 - binäre Suche 777, 780
 - binärer Suchbaum 780
 - Hashing 804
 - lineare Suche 775
- suchen 774
 - bsearch() 642
- suchen in einem String
 - strchr() 270
 - strcspn() 272
 - strpbrk() 276
 - strrchr() 276
 - strspn() 277
 - strstr() 277

switch-Anweisung 134
symbolische Konstante 203
system() 644
Systemprogrammierung 1091

T

tan() 626
Tangens 626
tanh() 627
Tastatur Puffer leeren 474
TCP 1031
TCP-Echo-Server 1054
 Client 1054
 Server 1057
telldir() 593
temporäre Datei
 mkstemp() 520
 tmpfile() 517
 tmpnam() 516
Tic Tac Toe 250
time() 608
time_t 607
TMP_MAX 516
tmpfile() 517
tmpnam() 516
tm-Struktur 607
toascii() 623
tolower() 623
toupper() 623
Trial-and-Error-Prinzip 833
Typendefinition
 typedef 424
Typenumwandlung
 explizit 115
 implizit 115

U

UCHAR_MAX 85
UDP 1031
UDP-Anwendung
 Client 1077
 Server 1076
UINT_MAX 86
ULONG_MAX 86
Umgebungsvariablen
 getenv() 914
Ungenauigkeit bei Gleitpunktzahlen
 76
ungetc() 472
union 416
UNIX 28
unsigned 83
UPDATE 968

URL 903
USE 962
USHRT_MAX 86

V

va_arg 597
va_end 598
va_list 597
va_start 597
variable Argumentliste bei Funktionen
 597
Variablen
 global 166
 Lebensdauer 169, 171
 lokal 163
 Speicherklassen 168
 statisch 167
 Typ-Qualifizierer 170
Variablen deklarieren 64
Vergleichsoperatoren 120
verkettete Listen
 doppelt 698
 einfach 671
 Queues 737
 Stack 715
Verzeichnis
 Arbeits- 589
 erstellen 583
 Funktionen 583
 lesen 591
 löschen 586
 positionieren 593
 wechseln 585
 zum Lesen öffnen 591
vfprintf() 605
void-Zeiger 331
volatile 170, 173
vprintf() 602
vsprintf() 605

W

Warteschlange 737
Webprogrammierung
 clientseitig 887
 serverseitig 887
Webserver
 Apache 891
 HTTPD 888
while-Schleife 138
Whitespace 59
Win32-Konsole 32
Winsock 1033
 initialisieren 1035

Wrapper-Funktion 872
write() 547
WSACleanup() 1035
WSAStartup() 1035

Z

Zeichen

- lesen 449, 451, 453
- schreiben 449, 453
- umwandeln in Großbuchstaben 623
- umwandeln in Kleinbuchstaben 623
- zurückschieben 472

Zeichenkette 259

Zeichenklassifizierung 623

Zeichenkonstanten 89

Zeichensatz

- international 82
- national 82
- OEM 83

Zeichensätze

- Basic-Zeichensatz 43
- Steuerzeichen 43

Zeiger 281

- als Rückgabewert 302
- auf Funktionen 325
- call by reference 298
- deklarieren 281
- Dereferenzierung 287
- die auf andere Zeiger verweisen 295
- dynamisches Zeigerarray 376

- initialisieren 282
- NULL 291
- ptrdiff_t 297
- read-only 314
- Speichergröße 294, 333
- void 331

Zeiger auf Strings 313

Zeiger auf Zeiger 315

Zeiger und Array 305, 334, 379

Zeigerarithmetik 295

zeilenweise Lesen 481

- bestimmte Zeile auslesen 488

- newline entfernen 483

- suchen nach Stringfolge 491

- suchen nach Wörter 492

- suchen und Ersetzen 493

- von Zeile x bis y lesen 489

zeilenweise Schreiben 485

Zeit 607

- tm-Struktur 607

- Zeitraum berechnen 614

Zeiten einer Datei

- Änderungszeit 571

- Inode-Änderung 571

- Zugriffszeit 571

Zufallszahlen

- rand() 638

- srand() 638

Zugriffsrechte 567, 580