

Inhaltsverzeichnis

0 Einleitung	3
0.1 Ein paar Worte zu C und seiner Geschichte	3
0.1.1 Die Entstehung von C	3
0.1.2 C als Vorstufe zu C++	3
0.1.3 Die Standardisierung von C	4
0.2 Hinweise zu diesem Buch und zum begleitenden Übungs- bzw. Lösungsbuch	5
0.3 Bezug aller Beispiel- und Übungsprogramme dieses Buches	6
1 Einführendes Beispiel	7
1.1 Erste wesentliche C-Regeln	7
1.2 Erstellen eines Programms (Bedienung)	8
1.3 Kompilieren eines Programms (Bedienung)	9
1.4 Starten eines kompilierten Programms (Bedienung)	10
1.5 Das Steuerzeichen \n	11
1.6 Zeilen-Kommentare mit // (neu in C99)	12
1.7 Tipps	12
1.8 Übungen	14
1.8.1 Ausgabe eines Menüs	14
1.8.2 Es weihnachtet sehr	14
1.8.3 Kommentar und Anführungszeichen	14
2 Elementare Datentypen	15
2.1 Positionssysteme	15
2.2 Negation von Zahlen durch Zweier-Komplement	16
2.3 Die Grunddatentypen in C	19
2.4 Wertebereiche für die einzelnen Datentypen	21
2.5 Fallgrube: Verlust von Bits bei zu grossen Zahlen	22
2.6 Übungen	23
2.6.1 Umwandlung von Dualzahlen in Dezimalzahlen	23
2.6.2 Umwandlung von Dezimalzahlen in Dualzahlen	23
2.6.3 Bereichsüberläufe beim Datentyp short	23
3 Konstanten	25

3.1	Oktale und Hexadezimale Zahlen	25
3.2	Verschiedene Arten von C-Konstanten	26
3.2.1	char-Konstanten	26
3.2.2	Ganzzahlige Konstanten	27
3.2.3	Gleitpunktkonstanten	27
3.3	Übungen	28
3.3.1	Bitmuster von Zeichen und Zahlen beim Datentyp char	28
3.3.2	Bitmuster für Oktal- und Hexazahlen beim Datentyp short	28
3.3.3	Erlaubte und unerlaubte Gleitpunktkonstanten	28
4	Variablen	29
4.1	Variablen und die C-Regeln für Variablennamen	29
4.2	Tipps zur Wahl der Variablennamen	31
4.2.1	Selbsterklärende Variablennamen	31
4.2.2	Unterstrich verbessert die Lesbarkeit	32
4.3	Deklaration von Variablen	32
4.4	Tipp: Variablen bereits bei Deklaration dokumentieren	34
4.5	Übung: Erlaubte und unerlaubte Variablennamen	34
5	Ausdrücke und Operatoren	35
5.1	Der einfache Zuweisungsoperator	35
5.1.1	Allgemeines zum einfachen Zuweisungsoperator	35
5.1.2	Initialisierung von Variablen	38
5.1.3	Übung: Zuweisen von unterschiedlichen Konstanten	38
5.2	Arithmetische Operatoren	39
5.2.1	Die arithmetischen Operatoren	39
5.2.2	Die C-Begriffe Ausdruck und Anweisung	40
5.2.3	Ausgabe von int-Variablen und -Ausdrücken	40
5.2.4	Ausgabe von Gleitpunkt-Variablen und -Ausdrücken	41
5.2.5	Fallgrube: Ganzzahl- statt Gleitpunktdivision	42
5.2.6	Übungen	43
5.3	Vergleichsoperatoren	44
5.3.1	Die unterschiedlichen Vergleichsoperatoren	44
5.3.2	Die zwei Wahrheitswerte von Vergleichen	44
5.3.3	Prioritäten der Vergleichsoperatoren	45
5.3.4	Übung: Prioritäten aller bisherigen Operatoren	45
5.4	Logische Operatoren	46
5.4.1	TRUE und FALSE in C	46
5.4.2	Der Datentyp _Bool (neu in C99)	46
5.4.3	Die C-Operatoren für NOT, AND und OR im Überblick	47
5.4.4	Der Negations-Operator !	47
5.4.5	Der AND-Operator &&	48
5.4.6	Der OR-Operator 	48
5.4.7	Beispiel zum neuen C99-Datentyp _Bool bzw. bool	49
5.4.8	Die Priorität der logischen Operatoren	49
5.4.9	Keine unnötige Auswertung rechts von && und 	50

5.4.10 Übung: Überprüfungen mit logischen Operatoren	51
5.5 Bit-Operatoren	52
5.5.1 Die Bit-Operatoren im Überblick	52
5.5.2 Bitweise Invertierung mit ~	52
5.5.3 Bitweise AND-Verknüpfung mit &	53
5.5.4 Bitweise OR-Verknüpfung mit 	55
5.5.5 Bitweise XOR-Verknüpfung mit ^	57
5.5.6 Bit-Operatoren nur für ganzzahlige Datentypen erlaubt	58
5.5.7 Fallgruben	59
5.5.8 Übung: Überprüfungen mit Bit-Operatoren	61
5.6 Shift-Operatoren	61
5.6.1 Die beiden Shift-Operatoren << und >>	61
5.6.2 Shift-Operatoren nur für ganzzahlige Datentypen erlaubt	63
5.6.3 Priorität der Shift-Operatoren	64
5.6.4 Übungen	64
5.7 Zusammengesetzte Zuweisungsoperatoren	65
5.7.1 Die zusammengesetzten Zuweisungsoperatoren	65
5.7.2 Übung zu den zusammengesetzten Operatoren	67
5.8 Inkrement- und Dekrement-Operatoren	68
5.8.1 Inkrementieren und Dekrementieren mit ++ und --	68
5.8.2 Präfix- und Postfix-Schreibweise für ++ und --	68
5.8.3 ++ und -- ist nur für Variablen erlaubt	70
5.8.4 ++ und -- ist nicht auf linken Seite einer Zuweisung erlaubt	70
5.8.5 Übung zu den Inkrement- und Dekrement-Operatoren	71
5.9 Priorität und Anwendbarkeit der Operatoren	71
5.9.1 Prioritätstabelle für Operatoren	71
5.9.2 Assoziativität der Operatoren	72
5.9.3 Erlaubte und unerlaubte Operationen für die C-Datentypen	73
5.9.4 Priorität und Auswertungszeitpunkt bei ++ und --	74
5.9.5 Fallgrube: Zugriff auf nicht vorbesetzte Variablen	75
5.9.6 Übungen	76
6 Symbolische Konstanten	77
6.1 Konstanten-Definition mit #define	77
6.1.1 Die Direktive #define	77
6.1.2 Regeln für Konstanten-Namen bei #define	79
6.1.3 Konstanten machen Programm leicht änderbar	79
6.2 Konstanten-Definition mit const	79
6.3 Übungen	80
6.3.1 Volumen und Oberfläche einer Kugel	80
6.3.2 Das Phänomen der entfesselten Erde	81
6.3.3 Benzinverbrauch und Durchschnitts-Geschwindigkeit	81
6.3.4 Geldscheine stapeln	82
7 Ein- und Ausgabe	83
7.1 Headerdateien und #include	83

7.1.1	Bibliotheken und Headerdateien	83
7.1.2	Eigene Headerdateien	85
7.2	Ein- und Ausgabe eines Zeichens	85
7.2.1	getchar() und putchar()	85
7.2.2	Gepufferte Eingabe bei getchar()	86
7.2.3	Puffer-Bereinigung mit Dummy-getchar()	89
7.2.4	Puffer-Bereinigung ist nicht immer notwendig	91
7.2.5	Fallgrube: Zahlen nicht mit getchar() einlesen	92
7.2.6	Die Headerdatei ctype.h	93
7.2.7	Einfache Makros	95
7.2.8	Übung: Umrechnung von Geschwindigkeiten	100
7.3	Die Ausgabe mit printf()	101
7.3.1	Die Funktion printf()	101
7.3.2	Fallgruben	108
7.3.3	Tipps	109
7.3.4	Übung: Die Capture-Recapture Methode	112
7.4	Die Eingabe mit scanf()	113
7.4.1	Die Funktion scanf()	113
7.4.2	Fallgruben	118
7.4.3	Die Headerdatei math.h	121
7.4.4	Fallgrube: Vergessen von #include <math.h>	127
7.4.5	Übungen	128
8	Datentypumwandlungen						131
8.1	Implizite Datentypumwandlungen	131
8.1.1	Der sizeof-Operator	131
8.1.2	Implizite Datentypumwandlungen	133
8.1.3	Fallgrube: Zuweisen von Ganzzahlausdrücken an Gleitpunktvariablen	140
8.1.4	Übung: Sparschweinhalt aufaddieren	141
8.2	Explizite Datentypumwandlungen	142
8.2.1	Explizite Datentypumwandlungen mit cast-Operator	142
8.2.2	Fallgruben	142
8.2.3	Übung: Prozentzahlen für die Kandidaten bei einer Wahl	145
9	Die Headerdateien limits.h und float.h						147
9.1	<limits.h> - Grenzwerte von Ganzzahltypen	147
9.1.1	Die Headerdatei limits.h	147
9.1.2	Übung: Wertebereiche der ganzzahligen Datentypen	148
9.2	<float.h> - Grenzwerte von Gleitpunkt-Datentypen	149
9.2.1	Das IEEE-Format	149
9.2.2	Die Headerdatei float.h	151
9.2.3	Umwandlung gebrochener Dezimalzahlen in das Dualsystem	152
9.2.4	Hexadezimale Aus-/Eingabe von Gleitpunktzahlen im IEEE-Format (neu in C99)	153

9.2.5 Übung: Eigenschaften von Gleitpunkttypen	155
10 Anweisungen und Blöcke	157
11 Die if-Anweisung	159
11.1 Die zweiseitige if-Anweisung	159
11.2 Die einseitige if-Anweisung	166
11.3 Verschachtelte if-Anweisungen	168
11.4 Tipp: Einrücken untergeordneter Programmteile	174
11.5 Fallgruben	174
11.5.1 Falsche Gleichheitsüberprüfung	174
11.5.2 Keine unnötige Auswertung rechts von && und 	175
11.5.3 Vergleiche von negativen Zahlen mit unsigned-Variablen	176
11.5.4 Hohe Priorität des Negations-Operators !	177
11.6 Programmiertechniken	178
11.6.1 if-Kaskaden	178
11.6.2 Pseudocode	180
11.7 Übungen	181
11.7.1 Schaltjahre (Struktogramm in C-Programm umformen)	181
11.7.2 Rechnungen erstellen	181
12 Die bedingte Bewertung	183
12.1 Die bedingte Bewertung ?:	183
12.2 Priorität des bedingten Operators	186
12.3 Tipp: Alternative Ausgaben bei printf()	187
12.4 Übung: Idealgewicht	187
13 Die switch-Anweisung	189
13.1 Die switch-Anweisung	189
13.2 Fallgrube: case-Marken müssen ganzzahlige Konstanten sein	196
13.3 Tipps	197
13.3.1 Alle case-Marken (auch letzte) mit break abschliessen	197
13.3.2 default immer angeben	197
13.4 Übung: Fläche, Umfang und Radius eines Kreises	198
14 Der Komma-Operator	199
14.1 Der Komma-Operator	199
14.2 Komma-Operator hat die niedrigste Priorität	200
14.3 Übung: Zusammenfassen mehrerer Anweisungen zu einer	200
15 Die for-Anweisung	201
15.1 Die for-Anweisung	201
15.2 Die for-Schleife und der Komma-Operator	207
15.3 Fallgrube: Semikolon am Ende des for-Schleifenkopfs	210
15.4 Geschachtelte Schleifen	211
15.5 Eine endlose for-Schleife	219
15.6 for bei Durchläufen mit festen Schrittweiten	219

15.7 Variablen Deklaration im for -Schleifenkopf (neu in C99)	222
15.8 Programmiertechniken	223
15.8.1 Anhalten einer Bildschirmausgabe	223
15.8.2 Zeilenvorschübe bei geschachtelten Schleifen	226
15.8.3 Kombinieren mit for -Schleifen	230
15.8.4 Zwischeninformationen bei rechenintensiven Programmen	233
15.8.5 Merker in for -Schleifen bei Eintreten von Ereignissen	235
15.9 Fallgruben	236
15.9.1 Niemals die Laufvariable im Schleifenkörper ändern	236
15.9.2 Gleitpunktzahlen niemals auf Gleichheit prüfen	240
15.9.3 Laufvariable einer for -Schleife läuft über Endwert hinaus	242
15.10 Übungen	243
15.10.1 Berechnung der harmonischen Reihe	243
15.10.2 Ausgabe der Dominosteine	243
16 Die while-Anweisung	245
16.1 Die while -Anweisung	245
16.2 Programmiertechniken	249
16.2.1 while bei unbekannter Zahl von Schleifendurchläufen	249
16.2.2 Konsistenzprüfungen bei Eingaben	251
16.2.3 Die Konstante EOF	252
16.2.4 Minimum und Maximum in einer Zahlenfolge	253
16.3 Zufallszahlen in C	254
16.4 Übungen	264
16.4.1 Fritz und Hans essen Äpfel	264
16.4.2 Primfaktor-Zerlegung	264
17 Die do...while-Anweisung	267
17.1 Die do...while -Anweisung	267
17.2 Programmiertechniken	270
17.2.1 do...while -Schleifen nicht so oft wie while -Schleifen	270
17.2.2 Abschließendes } while immer in einer Zeile	270
17.3 Übungen	271
17.3.1 Zahlen raten	271
17.3.2 Armstrong-Zahlen	272
18 Die break-Anweisung	273
18.1 Die break -Anweisung	273
18.2 break bewirkt Verlassen einer Schleifenebene	274
18.3 Programmiertechniken	275
18.3.1 Sofortiges Verlassen von Schleifen und switch	275
18.3.2 Endlosschleifen und break	277
18.4 Übungen	278
18.4.1 Würfelspiel bis 100	278
18.4.2 Primzahlen	278
19 Die continue-Anweisung	279

19.1	Die <code>continue</code> -Anweisung	279
19.2	Programmiertechniken	282
19.2.1	<code>continue</code> nur im äußersten Notfall	282
19.2.2	Korrekte Programme müssen auch schnell sein	283
19.3	Datums- und Zeitangaben (< <code>time.h</code> >)	286
19.3.1	Konstanten	286
19.3.2	Datentypen	287
19.3.3	Funktionen	287
19.3.4	Einige Funktionen aus <code>time.h</code> im Überblick	290
19.4	Übung: Drei Zahlen zu einer Summe finden	294
20	Marken und die <code>goto</code>-Anweisung	297
20.1	Marken und die <code>goto</code> -Anweisung	297
20.2	Programmiertechniken	298
20.2.1	<code>goto</code> nur im äußersten Notfall	298
20.2.2	Lesbarere und schnellere Programme mit <code>goto</code>	298
21	Graphikprogrammierung unter Linux	301
21.1	Bezug und Installation von LCGI	301
21.1.1	Bezug von LCGI	301
21.1.2	Installation von LCGI	302
21.2	Benutzung von LCGI	302
21.2.1	Inkludieren von < <code>graphics.h</code> >	302
21.2.2	Angabe von <code>main()</code> mit Parametern	302
21.2.3	Kompilieren und Linken von Graphikprogrammen	302
21.3	Einige für Graphik benötigte C-Konstrukte	303
21.3.1	Dynamisches Erstellen von Zeichenketten	303
21.3.2	Kurze Beschreibung von Arrays	304
21.4	Graphikmodus ein- und ausschalten	305
21.5	Eingaben im Graphikmodus	305
21.6	Bildschirm-, Farben- und Pixel-Operationen	310
21.7	Positionieren, Linien zeichnen und Farbe einstellen	314
21.8	Figuren zeichnen und ausfüllen	317
21.9	Einstellungen für Textausgaben	324
21.10	Externe Bilder laden, Bildteile speichern und einblenden	327
21.11	Kuchenstücke malen	330
21.12	Graphikpaket neu bzw. anders einrichten	332
21.13	Arbeiten mit mehreren Zeichenfenstern	333
21.14	Programmierung der Maus	334
21.15	Transformation mathematischer Koordinaten in das Graphikfenster	337
21.16	Kurzer Einblick in Fraktale und Chaos	341
21.17	Übungen	344
21.17.1	Ermitteln der Zahl PI mit Regentropfen	344
21.17.2	Basketball spielen	345
21.17.3	Die Kochsche Schneeflocke	346

21.17.4 Zeichnen und Füllen von Quadraten mit der Maus	347
22 Funktionen	349
22.1 Allgemeines zu Funktionen	349
22.1.1 Allgemeines Beispiel zu Funktionen	349
22.1.2 Die Begriffe Parameter und Argumente	351
22.1.3 Bibliotheken und Headerdateien	351
22.2 Erstellen eigener Funktionen	353
22.2.1 Definition von Funktionen in C89/C99	353
22.2.2 Definition von Funktionen in Alt-C	356
22.2.3 Die return-Anweisung	357
22.2.4 Funktionen ohne Rückgabewert	357
22.2.5 Forward-Deklarationen	359
22.2.6 Funktions-Prototypen	361
22.2.7 Implizite Datentypumwandlung beim Funktionsaufruf	366
22.2.8 Typische Anwendungsgebiete von Funktionen	369
22.2.9 Übungen	374
22.3 Die Parameter von Funktionen	376
22.3.1 Leere Parameterliste durch Angabe von void	376
22.3.2 Bei Funktionsaufrufen findet nur Wertübergabe statt	377
22.3.3 Call by reference	381
22.3.4 Auswertung der Argumente findet vor Funktionsaufruf statt	385
22.3.5 Fallgruben	386
22.3.6 Übung: Zeiten-Taschenrechner	388
22.4 Ellipsen-Prototypen für Funktionen mit variabler Argumentanzahl	389
22.4.1 Reihenfolge der Argument-Ablage im Stack	389
22.4.2 Ellipsen-Prototypen	389
22.4.3 Abarbeiten variabel langer Argumentlisten	389
22.4.4 Verfahren zum Abarbeiten variabel langer Argumentlisten	390
22.4.5 Fallgruben	395
22.4.6 Übungen	396
22.5 Neuheiten in C99	396
22.5.1 Inline-Funktionen	396
22.5.2 Der vordefinierte Name <code>__func__</code>	398
22.5.3 Keine Unterstützung von implizitem <code>int</code>	398
22.5.4 Keine impliziten Funktionsdeklarationen	399
22.5.5 Einschränkungen bei <code>return</code>	399
22.6 Rekursive Funktionen	399
22.6.1 Allgemeines zu rekursiven Funktionen	399
22.6.2 Einige typische Anwendungen für die Rekursion	404
22.6.3 Übungen	412
22.7 Zeiger auf Funktionen	415
22.7.1 Zeiger auf Funktionen	415
22.7.2 Typische Anwendungen	418
23 Speicherklassen und Modultechnik	423

23.1	Gültigkeitsbereich, Lebensdauer, Speicherort	423
23.1.1	Gültigkeitsbereich	423
23.1.2	Lebensdauer	431
23.1.3	Speicherort	431
23.1.4	Gültigkeit, Lebensdauer und Speicherort im Überblick	431
23.1.5	Übung: Ausgabe des Programms <code>block3.c</code>	432
23.2	Schlüsselwörter <code>extern</code> , <code>auto</code> , <code>static</code> und <code>register</code>	432
23.2.1	Das Schlüsselwort <code>extern</code>	432
23.2.2	Das Schlüsselwort <code>auto</code>	436
23.2.3	Fallgrube: Niemals Adressen von <code>auto</code> -Variablen zurückgeben	444
23.2.4	Das Schlüsselwort <code>static</code>	445
23.2.5	Das Schlüsselwort <code>register</code>	455
23.2.6	Übung: Ausgabe des Programms <code>speikla1.c</code>	456
23.3	Die Schlüsselwörter <code>const</code> und <code>volatile</code>	457
23.3.1	Das Schlüsselwort <code>const</code>	457
23.3.2	Das Schlüsselwort <code>volatile</code>	459
23.3.3	Kombination von <code>const</code> und <code>volatile</code>	460
23.3.4	Übung: Konstante Zeiger und Zeiger auf Konstanten	460
23.4	Die Modultechnik	461
23.4.1	Von der Maschinensprache bis zur Modultechnik	461
23.4.2	Modultechnik und Information Hiding	466
23.4.3	Software-Technik: Linker und Compiler	471
23.4.4	Beispiel: Simulation von Turingmaschinen	472
23.4.5	Übung: Turingmaschine zur binären Addition von 1	485
24	Präprozessor-Direktiven	487
24.1	Bedingte Kompilierung	488
24.1.1	Präprozessor-Direktiven zur bedingten Kompilierung	488
24.1.2	Typische Anwendungen	491
24.1.3	Testen mit Makro <code>assert()</code> aus Headerdatei <code><assert.h></code>	495
24.2	Einkopieren von anderen Headerdateien	497
24.2.1	Die Präprozessor-Direktive <code>#include</code>	497
24.2.2	Typische Anwendungen	498
24.3	Definition von Makros (<code>#define</code> und <code>#undef</code>)	499
24.3.1	Definition von Konstanten mit <code>#define</code>	500
24.3.2	Definition von Funktionsmakros mit <code>#define</code>	501
24.3.3	Operator <code>#</code> : Ersetzung von Makroparametern durch String	503
24.3.4	Operator <code>##</code> : Zusammensetzen neuer Namen	503
24.3.5	Rekursive Makrodefinitionen	504
24.3.6	Makros mit variabler Anzahl von Argumenten (neu in C99)	505
24.3.7	Makrodefinitionen mit <code>#undef</code> wieder aufheben	506
24.3.8	Unterschiede zwischen Funktionen und Makros	507
24.4	Vordefinierte Makronamen	511
24.5	Die restlichen Präprozessor-Direktiven	512
24.5.1	<code>#line</code> – Festlegen einer neuen Zeilennumerierung	512

24.5.2 #error – Ausgeben von Fehlermeldungen	513
24.5.3 #pragma – Festlegen von compilerspezifischem Verhalten	513
24.5.4 # – Die Null-Direktive	513
24.6 Übung: Ausgeben der Konstanten aus <code>limits.h</code> und <code>float.h</code>	514
25 Zeiger und Arrays	517
25.1 Eindimensionale Arrays	517
25.1.1 Eindimensionale Arrays	517
25.1.2 Nur statische Arrays erlaubt (in C89)	522
25.1.3 Von Arrays belegter Speicherplatz	522
25.1.4 Fallgruben	523
25.1.5 Übungen	525
25.2 Mehrdimensionale Arrays	527
25.2.1 Zweidimensionale Arrays	527
25.2.2 Drei-, vier-, fünf- und sonstige mehrdimensionale Arrays	536
25.2.3 <code>sizeof</code> liefert die Größe eines Arrays	536
25.2.4 Übung: Game of Life (Beispiel für zelluläre Automaten)	537
25.3 Zusammenhänge zwischen Arrays und Zeigern	539
25.3.1 Arrayname ist konstanter Zeiger auf erstes Element	539
25.3.2 Zugriff auf Arrayelemente ist auch über Zeiger möglich	542
25.3.3 Unterschied zwischen Arraynamen und echtem Zeiger	547
25.3.4 Erlaubte Operationen mit Zeigern	551
25.3.5 Unerlaubte Operationen mit Zeigern	552
25.3.6 Übergabe eines Arrays an eine Funktion mittels Adresse	555
25.3.7 <code>call by value</code> für Arrays (Zeiger)	559
25.3.8 Nachlese zu Arrays und Zeiger	560
25.3.9 Algorithmus: Der Bubble-Sort	560
25.3.10 Verwendung der Bibliotheksfunktion <code>qsort()</code>	562
25.3.11 Algorithmus: Binäre Suche	563
25.3.12 Verwendung der Bibliotheksfunktion <code>bsearch()</code>	566
25.3.13 Übungen	568
25.4 Strings und char-Zeiger	569
25.4.1 Besonderheiten von C-Strings	569
25.4.2 Das Schlüsselwort <code>restrict</code> für Zeiger (neu in C99)	571
25.4.3 Eigene Realisierung der Funktion <code>strcpy()</code> mit Arrays	572
25.4.4 Eigene Realisierung der Funktion <code>strcpy()</code> mit Zeigern	573
25.4.5 Die Headerdatei <code><string.h></code>	576
25.4.6 Umwandeln von Strings in numerische Werte	594
25.4.7 Umwandeln von numerischen Werten in Strings	601
25.4.8 Besonderheiten beim Einlesen von Strings mit <code>scanf()</code>	603
25.4.9 Ein- und Ausgabe von Strings mit <code>gets()</code> und <code>puts()</code>	604
25.4.10 Unterschied zwischen Zeiger- und Array-Deklaration	605
25.4.11 Direkter Zugriff auf Zeichen in einer String-Konstante	607
25.4.12 Übungen	608
25.5 Array-Initialisierungen	609
25.5.1 Initialisierung von Arrays	610

25.5.2 Dimensionierungsangaben bei der Initialisierung	613
25.5.3 Zeiger auf unbenannte Arrays (neu in C99)	615
25.5.4 Implizite Initialisierung bei static-Variablen/Arrays	615
25.5.5 Initialisierung lokaler Variablen auch mit Nicht-Konstanten	616
25.5.6 Initialisierung von lokalen Arrays in C89/C99	617
25.5.7 Initialisierung von lokalen Arrays mit variablen Werten (neu in C99)	618
25.5.8 Initialisierung von lokalen Arrays mit 0 oder NULL	619
25.5.9 Initialisierte Arrays mit const vor Überschreiben schützen	620
25.5.10 Übungen	620
25.6 Lokale Arrays variabler Länge (neu in C99)	622
25.7 Zeigerarrays und Zeiger auf Zeiger	623
25.7.1 Einfache Zeigerarrays	623
25.7.2 Zeiger auf Arrays	624
25.7.3 Vertauschen von zwei Arrays über Zeiger	625
25.7.4 Übergabe von Arrays an Funktionen	627
25.7.5 Zeiger-Zeiger	629
25.7.6 Unterschiede zwischen zweidimensionalen Arrays und Zeigerarrays	629
25.7.7 Zugriff auf beliebige Elemente in einem Zeigerarray	633
25.7.8 Zeigerarrays mit Funktionsadressen	639
25.7.9 Übungen	641
26 Argumente auf der Kommandozeile	645
26.1 Die Parameter argc und argv der Funktion main()	645
26.2 Optionen auf der Kommandozeile	648
26.3 Optionen auswerten mit der Funktion getopt()	655
26.4 Übung: Konvertieren von Dezimalzahlen in Dual, Oktal und Hexa	657
27 Dynamische Speicher-Reservierung und -Freigabe	659
27.1 Nachteile von statischen Arrays	659
27.1.1 Gefahr der Speicherüberschreibung	660
27.1.2 Speicherplatzvergeudung	661
27.2 Speicher reservieren mit malloc()	662
27.2.1 Die Funktion malloc()	662
27.2.2 Dynamische Arrays für beliebige Datentypen	667
27.2.3 Konvertierung von void-Zeigern	670
27.3 Speicher reservieren und initialisieren mit calloc()	671
27.4 Größenänderung eines allozierten Speicherbereichs mit realloc()	672
27.4.1 Die Funktion realloc()	672
27.4.2 Besonderheiten der Funktion realloc()	676
27.4.3 Schnellere Programme mit größeren Speicherblöcken	676
27.5 Freigeben eines dynamisch reservierten Speicherbereichs mit free()	679
27.5.1 Die Funktion free()	679

27.5.2 Fallgrube: <code>free()</code> setzt übergebenen Zeiger nicht auf NULL	679
27.5.3 Tipp: Eigenes Makro zur Freigabe von dynamischen Speicher	681
27.5.4 Fallgrube: <code>free()</code> nur auf von <code>malloc()</code> , <code>calloc()</code> und <code>realloc()</code> gelieferte Zeiger	681
27.6 Fallgrube: Allozieren von Speicherplatz in einer Funktion	683
27.7 Programmietechnik: Dynamische Zeiger-Arrays	687
27.8 Fallgrube: <code>free()</code> bei Zeiger-Arrays	688
27.9 Übung: Numerierte oder Rückwärtige Ausgabe eines Textes	688
28 Strukturen	689
28.1 Deklaration und Definition von Strukturen	689
28.1.1 Deklaration von Strukturen	689
28.1.2 Wichtige Regeln und Hinweise für Strukturdeklarationen	690
28.1.3 Definition von Strukturvariablen	691
28.1.4 Zusammenfassung von Strukturdeklaration und -definition	693
28.1.5 Namenlose Strukturen	694
28.2 Operationen mit Strukturvariablen	695
28.2.1 Zugriff auf Strukturkomponenten mittels Punktoperator	695
28.2.2 Zuweisung zwischen Strukturkomponenten	696
28.2.3 Zuweisung ganzer Strukturvariablen	701
28.2.4 Vergleich von Strukturvariablen ist nicht möglich	702
28.2.5 Casting für komplett Strukturvariable ist nicht möglich	702
28.2.6 Adreß- und <code>sizeof</code> -Operator für Strukturvariablen erlaubt	703
28.2.7 Übung: Bruchrechner	704
28.3 Initialisierung von Strukturvariablen	705
28.3.1 Initialisierung von Strukturvariablen in C89 und C99	705
28.3.2 Initialisierung von Strukturvariablen (nur in C99)	706
28.4 Strukturarrays	709
28.4.1 Arrays von Strukturvariablen	709
28.4.2 Übung: Zählen der Schlüsselwörter in einem C-Programm	719
28.5 Strukturen als Funktionsparameter	719
28.5.1 Übergabe von Strukturen an Funktionen	719
28.5.2 Übung: Tagesdifferenz zwischen zwei Daten	722
28.6 Zeiger und Strukturen	722
28.6.1 Allgemeines zu Zeiger und Strukturen	722
28.6.2 Dynamische Strukturarrays	737
28.6.3 Rekursive Strukturen	745
28.6.4 Übungen	779
28.7 Strukturen mit variabel langen Arrays (neu in C99)	782
28.8 Spezielle Strukturen (Unions und Bitfelder)	783
28.8.1 Unions	783
28.8.2 Bitfelder	788
29 Eigene Datentypen	795
29.1 Definition eigener Datentypnamen mit <code>typedef</code>	795

29.1.1	Vergabe von neuen Namen an existierende Datentypen mit <code>typedef</code>	795
29.1.2	Höhere Portabilität und bessere Lesbarkeit durch <code>typedef</code>	798
29.2	Definition eigener Datentypen mit <code>enum</code>	801
29.2.1	Definition eigener Datentypen mit <code>enum</code>	801
29.2.2	Regeln für <code>enum</code>	804
29.3	Übung: Mischtabelle aus der Chemie	805
30	Dateien	807
30.1	Höhere E/A-Funktionen	808
30.1.1	Vordefinierte Struktur <code>FILE</code>	808
30.1.2	Öffnen und Schließen von Dateien	809
30.1.3	Lesen und Schreiben in Dateien	813
30.1.4	Unterschied zwischen Text- und Binärmodus	837
30.1.5	Positionieren in Dateien	840
30.1.6	Öffnen einer Datei mit existierenden Stream	846
30.1.7	Löschen und Umbenennen von Dateien	848
30.1.8	Pufferung	848
30.1.9	Temporäre Dateien	851
30.1.10	Ausgabe von System-Fehlermeldungen	854
30.1.11	Übung: Ausgeben einer Datei mit Zeilennumerierung	859
30.2	Elementare E/A-Funktionen	859
30.2.1	Filedeskriptoren	860
30.2.2	Öffnen und Schließen von Dateien	860
30.2.3	Lesen und Schreiben in Dateien	863
30.2.4	Positionieren in Dateien	868
30.2.5	Effizienz von E/A-Operationen	870
30.2.6	Filedeskriptoren und der Datentyp <code>FILE</code>	872
30.2.7	Übung: Anhängen einer Datei an eine andere	873
31	Dateien, Directories und ihre Attribute	875
31.1	Dateiattribute	875
31.1.1	Struktur <code>stat</code> – Attribute zu einer Datei	875
31.1.2	<code>stat()</code> und <code>fstat()</code> – Erfragen von Dateiattributen	876
31.2	Dateiarten	877
31.3	Zugriffsrechte einer Datei	878
31.3.1	<code>chmod()</code> – Ändern der Zugriffsrechte für eine Datei	878
31.3.2	<code>access()</code> – Prüfen der Zugriffsrechte für eine Datei	879
31.4	Größe einer Datei	880
31.5	Zeiten einer Datei	881
31.6	Directories (Verzeichnisse)	882
31.6.1	<code>mkdir()</code> – Anlegen eines neuen Directory	882
31.6.2	<code>rmdir()</code> – Löschen eines leeren Directory	882
31.6.3	<code>chdir()</code> – Wechseln in ein neues Directory	882
31.6.4	<code>getcwd()</code> – Erfragen des Working-Directory-Pfadnamens	882

Inhaltsverzeichnis

31.6.5 <code>struct dirent</code> – Aufbau eines Eintrags in einer Directory- Datei	884
31.6.6 <code>opendir()</code> , <code>readdir()</code> , <code>rewinddir()</code> und <code>closedir()</code> – Lesen von Directories	884
31.6.7 Durchlaufen eines ganzen Directorybaums	886
31.7 Gerätedateien	890
31.8 Übung: Ermitteln der Bytes eines Directory	892
32 Die Umgebung eines ablaufenden Programms	893
32.1 Start eines Prozesses	893
32.1.1 Startup-Routine - Startadresse eines Programms	894
32.1.2 <code>main()</code> – Benutzerdefinierter Startpunkt eines Programms	894
32.2 Beendigung eines Prozesses	894
32.2.1 Exit-Status eines Prozesses	894
32.2.2 Normales Beenden der Funktion <code>main()</code> mit <code>return</code>	897
32.2.3 <code>exit()</code> – Normales Beenden eines Programms mit <code>cleanup</code>	897
32.2.4 <code>_exit()</code> – Beenden eines Programms ohne <code>cleanup</code>	897
32.2.5 <code>atexit()</code> – Einrichten von Exithandlern	898
32.2.6 Start und Beendigung eines Prozesses im Überblick	899
32.3 Environment eines Prozesses	900
32.3.1 Environment-Liste	900
32.3.2 Zugriff auf die ganze Environment-Liste	900
32.3.3 <code>getenv()</code> – Erfragen einzelner Environment-Variablen	902
32.3.4 <code>putenv()</code> , <code>setenv()</code> und <code>unsetenv()</code> – Ändern, Hinzufügen oder Löschen von Environment-Variablen	902
32.3.5 Übung: Realisierung des <code>which</code> -Kommandos	903
33 Starten eines anderen Programms	905
33.1 Die Funktion <code>system()</code>	905
33.2 Die <code>exec()</code> -Funktionen	906
33.2.1 Unterschiede der <code>exec()</code> -Funktionen im Überblick	907
33.2.2 Interpretation des Dateinamens bei <code>execvp()</code> und <code>execv()</code>	908
33.2.3 Unterschiede in der Form der Argumentübergabe	908
33.2.4 Unterschiede bei Benutzung der Environment	908
33.2.5 Vererbungen bei <code>exec()</code>	909
33.3 Übung: Interaktiver Directory-Wechsel	910
34 Signale	913
34.1 Das Signalkonzept und die Funktion <code>signal()</code>	913
34.1.1 <code>signal()</code> – Einrichten von Signalhandlern	914
34.1.2 Signale und die exec-Funktionen	916
34.2 Signallisten und Signallnummern	916
34.3 Mögliche Probleme mit der <code>signal()</code> -Funktion	917
34.3.1 Erfragen des aktuellen Signalstatus ohne Änderung nicht möglich	
.	918

34.3.2 Zeitspanne zwischen Auftreten eines Signals und Aufruf der signal() -Funktion	918
34.3.3 Endlosschleifen beim Warten auf das Eintreten von Signalen	919
34.4 Anormale Beendigung mit Funktion abort ()	920
34.5 Anhalten eines Prozesses mit Funktion sleep ()	920
34.6 Übung: Reaktionstest über Signale	920
35 Nicht-Lokale Sprünge	923
35.1 setjmp () und longjmp () – Springen über Funktionsgrenzen	923
35.1.1 Der Datentyp jmp_buf	924
35.1.2 setjmp () – Einrichten eines Ansprungpunktes	925
35.1.3 longjmp () – Sprung zu einem mit setjmp () markierten Punkt	925
35.2 Rückkehr über mehrere Ebenen bei bestimmten Ereignissen	925
35.3 auto-, register-, static- und volatile-Variable bei longjmp ()	931
36 Weitere Headerdateien	933
36.1 Die Headerdateien von C89 und C99 im Überblick	933
36.2 Die Headerdatei <locale.h>	935
36.2.1 Allgemeines zur Headerdatei <locale.h>	935
36.2.2 Die Funktion setlocale()	936
36.2.3 Die Funktion localeconv()	937
36.3 Wide-Character-Funktionen	942
36.3.1 Wide-Character-Klassifizierungsfunktionen	943
36.3.2 Wide-Character-E/A-Funktionen	945
36.3.3 Wide-Character-Stringfunktionen	946
36.3.4 Wide-Character-Speicherfunktionen	947
36.3.5 Wide-Character-String-Konvertierungsfunktionen	948
36.3.6 Multibyte/Wide-Character-Konvertierungsfunktionen	948
36.4 Die Headerdatei <complex.h> (neu in C99)	951
36.5 Die Headerdatei <fenv.h> (neu in C99)	955
36.6 Die Headerdatei <stdint.h> (neu in C99)	958
36.7 Die Headerdatei <inttypes.h> (neu in C99)	959
36.8 Die Headerdatei <tgmath.h> (neu in C99)	964
36.9 Die Headerdatei <iso646.h> (neu in C99)	965
37 Lösungen zu den Übungen	967
37.1 Lösungen zu Kapitel 1	967
37.2 Lösungen zu Kapitel 2	968
37.3 Lösungen zu Kapitel 3	972
37.4 Lösungen zu Kapitel 4	973
37.5 Lösungen zu Kapitel 5	973
37.6 Lösungen zu Kapitel 6	979
37.7 Lösungen zu Kapitel 7	981
37.8 Lösungen zu Kapitel 8	985
37.9 Lösungen zu Kapitel 9	986

37.10 Lösungen zu Kapitel 10	988
37.11 Lösungen zu Kapitel 11	988
37.12 Lösungen zu Kapitel 12	990
37.13 Lösungen zu Kapitel 13	991
37.14 Lösungen zu Kapitel 14	992
37.15 Lösungen zu Kapitel 15	992
37.16 Lösungen zu Kapitel 16	993
37.17 Lösungen zu Kapitel 17	995
37.18 Lösungen zu Kapitel 18	996
37.19 Lösungen zu Kapitel 19	999
37.20 Lösungen zu Kapitel 20	1002
37.21 Lösungen zu Kapitel 21	1002
37.22 Lösungen zu Kapitel 22	1010
37.23 Lösungen zu Kapitel 23	1019
37.24 Lösungen zu Kapitel 24	1023
37.25 Lösungen zu Kapitel 25	1025
37.26 Lösungen zu Kapitel 26	1038
37.27 Lösungen zu Kapitel 27	1040
37.28 Lösungen zu Kapitel 28	1041
37.29 Lösungen zu Kapitel 29	1047
37.30 Lösungen zu Kapitel 30	1048
37.31 Lösungen zu Kapitel 31	1050
37.32 Lösungen zu Kapitel 32	1052
37.33 Lösungen zu Kapitel 33	1053
37.34 Lösungen zu Kapitel 34	1054
38 Anhang	1057
38.1 Wichtige Tabellen zum Nachschlagen	1057
38.1.1 Prioritätstabelle für die Operatoren	1057
38.1.2 Wertebereiche für die einzelnen Datentypen	1058
38.1.3 Die Funktion <code>printf()</code>	1059
38.1.4 Die Funktion <code>scanf()</code>	1061
38.2 Dezimal-, Hexa-, Oktal- und Dualtabelle	1063
38.3 ASCII-Tabelle	1065
38.4 Literaturverzeichnis	1066
38.5 Bestelldaten für Übungen und Lösungen zu diesem Buch	1068