

# Inhaltsverzeichnis

<b>Einführung</b>	<b>23</b>
Was bringt es, C zu lernen?	23
Über dieses Buch	24
Programme in diesem Buch	24
Törichte Annahmen über den Leser	25
Wie dieses Buch aufgebaut ist	26
Teil I: Das erste Programm	26
Teil II: Grundlegendes Sprachverständnis	26
Teil III: Einfache Datenstrukturen und Zeiger	26
Teil IV: Daten speichern und verwalten	26
Teil V: Der Top-Ten-Teil	26
Symbole, die in diesem Buch verwendet werden	27
Schlussgedanken	27
<b>Teil I</b>	
<b>Das erste Programm</b>	<b>29</b>
<b>Kapitel 1</b>	
<b>Der (zumeist harmlose) Einstieg</b>	<b>31</b>
Eine extrem kurze und banale Geschichte über C	31
Wie aus einer süßen kleinen Textdatei ein Programm wird	32
Der Entwicklungsprozess in C	33
Der Quelltext (oder auch Sourcecode)	34
Der Compiler	35
Der Linker	36
Das erste C-Programm – eine alte Tradition	37
Speichern! Kompilieren! Linken! Starten!	39
Die notwendigen Editier- und Kompiliekünste	39
Die Quelltextdatei ändern	40
Neukompilierung (oder: Spiel's noch mal in der Sprache C)	41
»Aber mein Compiler kompiliert's nicht neu!«	41
Raus mit den alten Sachen, lasst uns was Neues machen	42
Mit Fehlern auf Du und Du	42
Mist! Ein Fehler. Aber bevor Sie aus dem Fenster springen ...	42
Junge, was'n Fehler!	44
Die Fehlermeldung untersuchen	44
Das fehlerhafte Programm reparieren	46

# *C für Dummies*

Die fürchterlichen Linkerfehler	46
Linkerfehler beheben	47
Wie die Sprache C aussieht	48
Der große Zusammenhang	49
Stückliste	50
Darf ich vorstellen? Die Sprache C mit ihren Schlüsselwörtern	51
Andere Sprachelemente von C	52
Eingaben und Ausgaben (die hier mal was Gutes sind)	53
Stellen Sie sich bei Herrn Computer vor	53
Die Belohnung!	54
Erst das Chaos, dann die Ordnung	56
Der C-Compiler räumt sein Zimmer nicht auf	56
Was aufheben, was wegwerfen?	57
Organisation ist alles!	57
Wichtige C-Regeln, an die Sie nie denken werden	58
Das hilfreiche Regelprogramm	58
Zeit für eine Bonusrunde	59

## *Kapitel 2*

### *Einfache C-Programme basteln*

**61**

printf (wer bei print an drucken denkt, liegt falsch)	61
Ein anderes »printf macht was Lustiges«-Programm	62
Noch mehr lustige Texte	63
Einige Escape-Sequenzen	64
Das f steht für formatiert	65
Richt' Euch!	66
scanf – lassen Sie Ihren Scanner trotzdem aus!	68
scanf richtig einsetzen	69
scanf im scharfen Einsatz	71
Kleine Änderungen an Whoru.c	72
Das Wunder von %s	72
Zeit für Experimente	73
Bemerkungen, Kommentare und Vorschläge	75
Darf ich um einen kurzen Kommentar bitten?	75
Kommentarstile der Ausgebufften und der weniger Ausgebufften	75
Wieso sind Kommentare notwendig?	76
Bizzare Kommentare	77
Kommentare als Schalter	78
Vermeidung »verschachtelter« Kommentare	79
gets, fgets und puts	80
Und tschüss scanf, willkommen gets	81
Ein unfreundliches Programmbeispiel	81
Das finstere Geheimnis der Sprache C	82
Sichere Dateneingaben	83
put – put – putputput – puts	83

## **Inhaltsverzeichnis**

Noch eine doofe Spielerei	84
puts und Variablen	85
Licht! Kamera! Action! puts und fgets die Zweite	85
Mehr Spaß mit printf	87
Das alte »Text mit printf ausgeben«	87
Die Escape-Sequenzen von printf	88
Das de luxe-Testprogramm für Escape-Sequenzen	88
Weitere Tests mit Printfun	90
Wieder mal die Komplexität des printf-Formats	91
Die Konvertierungszeichen von printf	92
 <b>Kapitel 3</b>	
<b>Variablen und Mathe – kein Grund, gleich in Tränen auszubrechen</b>	<b>95</b>
Die sich ständig ändernde Variable	95
Strings ändern sich	96
Willkommen in der Welt der numerischen Variablen	97
Benutzung der ersten Integervariablen	98
Zuweisung von Werten an numerische Variablen	99
Die Eingabe von Zahlen über die Tastatur	100
Wie alt war Methusalem gleich noch mal?	101
Mehr numerische Variablen und ein bisschen Mathe	104
Wieder auf der Jagd	104
Eine Portion Mathe	105
Wie lange brauchen Sie noch, um Methusalems Rekord zu brechen?	106
Methusalem: Bonusrunde! Jedes Los ein Gewinn!	107
Das unmittelbare Ergebnis	108
Diskurse, Diskussionen und die Deklaration von Variablen	109
»Warum muss ich eine Variable deklarieren?«	110
Verbotene und erlaubte Variablennamen	111
Vordefinieren von Variablen	111
Noch ein x-beliebiges Beispiel für Variablen	113
Mehrfachdeklarationen	114
Konstanten und Variablen	115
Träume von Freiheit und mehr Konstanz	115
Eine praktische Abkürzung	116
Das Schlüsselwort const	117
Der dritte Weg	118
Malen mit Zahlen	120
Zahlen in C	120
Wieso nimmt man int?	121
Mit oder ohne Vorzeichen – das ist hier die Frage	122
Wie bringt man eine Zahl eigentlich zum Fließen?	123
»Hey EIX32A, lass uns mal ein Fließkommaprogramm schreiben!«	124
Die Sache mit der E-Notation	125
Doppelt so groß wie float? Das ist double!	126

## **C für Dummies**

Bringen Sie Ihren Zahlen Format bei!	127
Die etwas andere Variable: char	128
Variablen mit einzelnen Zeichen	128
Variablen mit Zeichen abfüllen	130
Kitzelt eine Tasteneingabe den Computer?	130
Zeichenvariablen als Werte	132
Die erste wirkliche Mathestunde	133
Ein sehr knapper Überblick über die Operatoren	133
Das alte »Wie groß bist du?«-Programm	134
Heimtückische Änderungen an unserer Größenberechnung	135
Die hohe Kunst des Inkrementierens (»Zähl's dazu, Sam«)	136
Weniger lustig: Gewichtszunahme	137
Bonusprogramm! (Das kann Ihnen vielleicht wirklich mal was nützen)	138
Wer hat den Vortritt?	139
Ein Beispiel aus einer Examensklausur	140
Punktrechnung vor Strichrechnung	140
Klammern haben den Vortritt	141

## **Teil II**

### **Grundlegendes Sprachverständnis**

**145**

#### **Kapitel 4**

##### **Wir stehen vor einer großen Entscheidung**

**147**

Die mächtige if-Anweisung	147
Das Beispiel mit dem Computer als Genie	148
Das Schlüsselwort if ganz aus der Nähe	150
Eine Anmerkung zur Schreibweise der if-Anweisung	154
Die endgültige Lösung für die Steuerproblematik	155
Gut, wenn's nicht wahr ist, was ist es dann?	156
Mit einem else alle Möglichkeiten abdecken	157
Eine Anmerkung zur Schreibweise	159
Der Sonderfall else-if und noch viel mehr Entscheidungen	160
if mit Zeichen und Strings	161
Die zahlenlose Welt von if	162
Was ist größer: S oder T, \$ oder -?	162
if und der Vergleich von Strings	164
Wie schreibt man richtig in C? Lektion 1	164
Immer von oben nach unten	164
Formatierungen des Quellcodes	165
Einrückungen, Teil 2	166
Vergessen wir mal kurz die Formatierung	166
Andere Möglichkeiten	167
for knüpft Schleifen	168

## **Inhaltsverzeichnis**

Wiederholte Wiederholungen	169
for formt die Schleifen	171
Wie in der Grundschule: Zählen bis 100	173
Schleifen fabrizieren	174
Endlich – das nützliche ASCII-Programm	175
Vorsicht Endlosschleifen!	176
Schleifen gewaltsam abbrechen	177
Das Schlüsselwort break	179
Abkürzungen und die Kunst des Inkrementierens	179
Kryptische C-Operatoren, Teil 1: ++	180
Bahn frei, jetzt komm ich!	181
Rückwärts zählen: nelhäz sträwkcüR	182
Rückwärts zählen passt sehr gut zum for	183
Kryptische C-Operatoren, Teil 2: --	183
Eine abschließende Verbesserung von Ollyolly.c	184
Je mehr Inkrement, desto mehr Wahnsinn	184
Schleifen mit Sprüngen	185
Kryptische C-Operatoren, Teil 3: Jetzt wird's verrückt	186

## **Kapitel 5**

### **Ihre ganz persönlichen Funktionen**

	<b>189</b>
Die allererste Funktion schreiben	189
Dieses Programm braucht eine Funktion	191
Die nützliche Funktion idiot()	191
Das Tao der Funktionen	192
Etwas zur Namensgebung von Funktionen	193
Ein Wort zum Prototyping	194
Variablen innerhalb von Funktionen	196
Bomben frei mit dem Bomber-Programm	196
Bringt die doppelte Variable Bomber.c zur Explosion?	197
Ein weiterführendes Beispiel	198
Brüderliches Teilen von Werten mithilfe globaler Variablen	200
Eine globale Variable erzeugen	201
Ein Beispiel für eine globale Variable	202
Funktionen ein Päckchen mit auf den Weg geben	204
Funktionen mit einer echten Funktion	206
Wie man einer Funktion einen Wert schickt	206
Variablenwirrwarr vermeiden (unbedingt lesen)	208
Mehr als einen Wert an eine Funktion übergeben	208
Wie man Strings an Funktionen übergibt	210
Etwas mehr Ärger für Sie (Funktionen mit Ergebnissen)	212
Endlich sagt der Computer mal was Gutes über Sie	213
Fehlerbeseitigung in Iq.c mit dem beliebten Typecasting-Trick	214
Mit return zurück an den Absender	216
Gib dem Mann mal einen Bonus!	217

## **C für Dummies**

Funktionen arbeiten von innen nach außen	219
Die Sprache C arbeitet von innen nach außen	220
Veränderungen am Programm Iq.c	221
Ein gängiges Beispiel für if	222
Das alte Spiel mit den Zufallszahlen	223
Benutzung der rand-Funktion	224
Zufallszahlen initialisieren	226
Mehr Zufall im Zufallsprogramm	227
Verbesserungen am Zufallszahlenprogramm	228
Der teuflische Dr. Modulus	230
Die Würfel sind gefallen	231

## **Kapitel 6**

### **Feinschliff für die C-Künste**

**235**

Dieses Zeug drängelt sich immer vor (der Rest folgt dem #)	235
Bitte vergiss mein nicht!	236
Ihre eigene h-Datei erstellen	238
Wofür #define gut ist	240
Makros erwähnen wir besser erst gar nicht	241
Ich kann dir Geschichten erzählen (wie man Eingaben vom Benutzer bekommt)	243
Mit scanf die Tastatur abhören	243
Das fürchterliche Ratespiel	246
Die Shellbeklauen (wie man die Kommandozeile abhört)	248
Da gehen einem die Argumente aus – main kann das nie passieren	249
Also, irgendwelche Argumente dafür?	250
Anzeige der Argumente	251
Wiederholungen sind gut	254
Ihr Gewicht auf dem Mond	255
Noch mehr ifs	257
Auswertung von mehreren Bedingungen in if-Abfragen	258
Wer ist schlauer? Melvin oder Poindexter?	260
Gutes p, schlechtes P	262
Weitere Weisheiten	263
Nun zu den schlechten Nachrichten	263
Mehr über Variablen	264
Typecasting und andere Probleme	264
Grundlagen der Typumwandlung	265
Die lange und die kurze Form	265
Die Konfusion weicht allmählich	267
signed und unsigned	267
unsigned Variablen haben eine negative Einstellung	268
faire und unfaire Variablen	269
Ein anderes Kapitel über Zahlen	
(die Sache mit der Hexerei)	270
Zahlensysteme und ihre Basis	270

## **Inhaltsverzeichnis**

Hexereien	271
Darstellung von Zahlen mal anders	272
<b>Kapitel 7</b>	
<b>Schleifen knüpfen</b>	<b>275</b>
Fakten zu while-Schleifen	275
Die erste Schleife binden	276
Das Schlüsselwort while	277
Was ist besser geeignet: while oder for?	279
Tausch des unschönen for(;;) gegen ein elegantes while	280
Den Spieß umgedreht: do-while-Schleifen	281
while macht Kopfstände – do-while	282
De-tail von do-while	284
Trau niemals dem User – ein Fehler in Countdown.c	285
Die garantiert richtige Eingabe	286
Der bizarre Fall while TRUE	
(und sonstige Kuriositäten)	287
Fragestunde	287
Feintuning für Yorn.c	288
»Kochen, bis es gar ist«	289
Yorn.c als Funktion	291
Verschachtelte Schleifen und anderer Unsinn für Angeber	292
Verschachtelte Anwendungen	293
Schleifen und das heimtückische switch-case	295
Die Lösung mit switch-case	296
Der alte Trick mit switch-case	298
Eine besondere Beziehung zwischen while und switch-case	302
<b>Kapitel 8</b>	
<b>Zwischenstand und Reste essen</b>	<b>305</b>
Selbsttest via Hallo, Welt!	305
Das Hallo-Welt-Programm	306
Weiter geht's – ein närrisches Hallo-Welt-Programm	306
Halt die Welt an, ich möchte aussteigen!	307
Die Behandlung der Überbleibsel	310
Nun die Lösung mittels while-Schleife	312
Mehr über die Math.h-Library	313
Die Probleme in einer höheren Potenz	314
Autsch! Wurzeln ziehen	317
Lästige Mathematik? – Sie packen das	318
Etwas wahrhaft Merkwürdiges zum Schluss	319
Die Gefahren der Benutzung von a++	320
Ja, und das Gleiche gilt für --	321
Wiederholung des ++a-Phänomens	322

# **C für Dummies**

goto-Anweisung – nein danke	322
Was ist nun goto?	323
Ein Beispiel, wie man goto nicht verwenden sollte	324
Alternativen zu goto	326
und tschüss ... – das Programm verlassen	328
exit – der Notausgang	328
Zuletzt das berüchtigte Ask-Programm	329
Eine sinnlose Batchdatei als Beispiel	332
<b>Teil III</b>	
<b>Einfache Datenstrukturen und Zeiger</b>	<b>333</b>
<b>Kapitel 9</b>	
<b>Arrays und Strings</b>	<b>335</b>
Wozu Arrays?	335
Wie man Arrays benutzt	336
Ein einfaches Programm, bevor es zu langweilig wird	337
Arrays in C erstellen	338
Auf die Elemente eines Arrays zugreifen	340
Werte an ein Array zuweisen	341
Alt, aber brauchbar	343
Strings und Arrays	346
Das Ende eines Strings (das Nullbyte)	347
Köder auslegen	348
Die Falle	350
Gegen den Buffer-Overflow	352
Arrays jeder Sorte	353
Sortier mich, aber schnell	353
Sortieren	355
Eine unverschämte große Menge Zahlen sortieren	358
<b>Kapitel 10</b>	
<b>Strings und so Zeugs</b>	<b>363</b>
Strings und Zeichen	363
Das Programm »Mach mich GROSS«	364
Eine bessere Möglichkeit, einen String zu durchlaufen	366
Ein total verdrehter String	367
Texte verflechten	369
Strings in C kopieren	371
Strings zerlegen	373
Sesam öffne dich	376
Stringmanipulationen – ein wahres Tollhaus	378

## *Inhaltsverzeichnis*

Strings verbinden	378
Die Stringfunktionen zusammengefasst	381
Arrays jenseits der ersten Dimension	382
Vereinbarung eines zweidimensionalen Arrays	382
Vereinbarung eines initialisierten zweidimensionalen Arrays	383
Zugriff auf die Elemente eines zweidimensionalen Arrays	385
Ein Array aus Strings	386
Arrays in der Grauzone	391
Dreidimensionale String-Arrays	392
Wieder hinter den sieben Bergen bei den sieben Zwergen	393

### *Kapitel 11*

#### *Die finstersten Aspekte von C: Pointer*

**397**

Ein unhandliches und starres Speicherkonzept	397
Land kaufen, aber kein Haus bauen	398
Mehr oder weniger gefräßige Variablen	399
Die Größe eines Arrays berechnen	402
Lange Rede, kurzer Sinn	403
Adresse, Adresse, Adresse	403
Ein etwas simples Programm als brauchbares Beispiel	404
Hallo, Herr Zeiger!	405
Das zweite Metric-Programm	407
Die Vereinbarung der Pointer-Variablen	407
Pointer zum Speichern von Adressen benutzen	408
Was weiß nun das Programm?	409
Oh Schreck – eine kleine Übung zur Selbstkontrolle	410
Nun die Adresse der letzten Variablen	411
Mehr Pointer, mehr Speicher, Wahnsinn ohne Ende	413
Römische Zahlen (und wieder eine Ausnahme)	415
Noch mehr Römer	417
Das total verdorbene Römer-Programm	418
Und nun das Sternchen, bitte schön	419
Rückblick	419
Pointer, die zeigen und schauen	420
Der Pointer als neugieriger Postbote	421
Noch mehr Durcheinander mit den *-Pointern	422
Ein Pointer und mehrere Variablen	424
Pointer und Arrays	425
Ein Array mit einem Pointer durchlaufen	425
Nun eine wahrhaft schwer zu knackende Nuss	427
Pointer, Klammern, Arithmetik – puh	428
Pointer sollen Arrays nicht ersetzen	429
Zeit, einen Schnitt zu machen	431
Die schwierige Beziehung zwischen Pointern und Array-Klammern	431
Beseitigung aller Spuren der Array-Notation	432

## **C für Dummies**

Ungläubige können ja nachprüfen	433
Weitere Prüfung	433
Zum guten Schluss: Der Hauptunterschied zwischen Pointern und Arrays	434

### **Kapitel 12**

#### **Erste Anwendungen der Pointer**

**435**

Pointer und Strings	435
Eine ganz unmögliche Art, einen String auszugeben	435
Eine bessere Möglichkeit, einen String anzuzeigen	436
Mehr Tricks	438
Noch mehr Tricks	438
Ein allerletzter Trick	439
Die Tiefen der Bibliotheksdefinitionen	439
Oh! String! String! String!	440
Noch so eine irreführende Sache	441
Vereinbarung eines Strings mit einem char-Pointer	443
Übergabe von Pointern an Funktionen	444
Rückblick zu Pointer-Problemen	445
Botschaften aus dem Jenseits	447
Übergabe von Arrays an und von Funktionen	451
Übergabe eines char-Arrays an eine Funktion	452
Die verbotenen Vokale	454
Übergabe beliebiger Arrays an Funktionen	456
Analyse der caffeine-Funktion	460
Der Horror vor Funktionen, die ihre Variablen töten (und wie man es verhindert)	460
Arrays aus Pointern	462
Pointer und Arrays	462
Ein Stringpointer-Array erzeugen	463
Ein Beispielprogramm	464
Pointer-Array der Kommandozeile	466
Was zum Kuckuck ist das nun?	467
Pointer (auch in Arrays) sind Variablen	469
Sortieren von Strings mit Pointern	470
Ein einfaches, noch nicht ganz richtiges Sortierprogramm	470
Was war falsch?	472
Die entsetzlichen Indirektpointer	472
Sortieren von Strings nach allen Zeichen	475

### **Kapitel 13**

#### **Alles über Strukturen**

**479**

Mehrfachvariablen	479
Das Leben ohne Strukturen	480
Mit Arrays geht es nicht viel besser	482

## ***Inhaltsverzeichnis***

Wir brauchen eine Karteikarte	482
Fakten zu Strukturen	484
Noch einmal das Passwort-Programm	487
Arrays aus Strukturen	490
Alles in einem Array	492
Strukturkomponenten kopieren	495
Strukturen und Funktionen	500
Ein spezieller Rückblick: Vereinbarung einer globalen Variablen	500
Hey, Funktion, hier kommt eine Struktur!	501
Verschachtelte Strukturen	503
Rückgabe einer Struktur aus einer Funktion	506
Großer Dank gilt der Funktion malloc	511
Ein einfaches Beispiel ohne zusätzlichen Speicherplatz	512
Oh malloc, gib mir mehr Platz	513
Borgen und nicht wiedergeben ist wie gestohlen	515
Nun die richtige Speicherbereitstellung für Howdy.c	516
Strukturen und (schon wieder!) Pointer	518
Was Sie brauchen, um eine Struktur im Speicher anzulegen	519
Das erste Programm mit Pointer und Struktur	520

## ***Teil IV***

### ***Daten speichern und verwalten***

**523**

#### ***Kapitel 14***

##### ***Die Festplatte als Diener***

**525**

Hello Disk!	525
Ein kleines Textstück auf die Platte schreiben	526
Wie die Arbeit mit Dateien funktioniert	527
Etwas aus einer Datei lesen	530
Versehentliches Überschreiben verhindern	531
Binär oder Text – das ist hier die Frage	534
Ein eigenes type-Programm	535
Ein Dump erstellen	538
Formatierte Ein-/Ausgabe	540
Die formatierte Ausgabe	540
Formatierte Eingabe aus Dateien	541
Ein Array in eine Datei schreiben	542
Ein Array aus der Datei lesen	543
Daten lesen und schreiben	545
Das Grundgerüst für ein Programm zum Speichern von Strukturen	545
Eine Struktur in eine Datei schreiben	547
Die Funktion write_info	548
Die read_info-Funktion	550

# *C für Dummies*

<b>Kapitel 15</b>		
<b>Warum verkettete Listen?</b>		<b>555</b>
Kurzer Rückblick zur Datenspeicherung in C	555	
Wie verkettete Listen arbeiten	556	
Die erste Struktur	558	
Eine Struktur mehr und der Link darauf	559	
Und nun der Rest	561	
Die NULL kennzeichnet das Ende	563	
<b>Kapitel 16</b>		
<b>Die Geburt unserer Datenbank</b>		<b>567</b>
Das unvermeidliche Bankkontenprogramm	567	
Das BANK-Programm	567	
Was bereits funktioniert	571	
Datensätze aus Listen löschen	572	
Überflüssige Datensätze entfernen	573	
Die Funktion deleteAccount()	574	
Die Arbeitsweise der Funktion deleteAccount	576	
Einmal vom Speicher zur Disk und zurück	578	
Die Liste auf Platte speichern	578	
Eine verkettete Liste von Platte laden	579	
<b>Teil V</b>		
<b>Der Top-Ten-Teil</b>		<b>583</b>
<b>Kapitel 17</b>		
<b>Cn Gründe für C im Jahr 2010, 2011, 2012 ...</b>		<b>585</b>
C++, C#, Java, PHP und die anderen {}	585	
C ist schlank	585	
C ist fahren ohne ABS und ESP	585	
C ist die Sprache der APIs und Kernel	586	
C ist 31337	586	
C ist die Sprache der Aufzüge und Kühlschränke	586	
C ist allgegenwärtig	587	
C ist die Sprache der Schnittstellen	587	
C treibt die Dinge voran	587	
C ist wichtig ...	588	
<b>Kapitel 18</b>		
<b>Zehn Empfehlungen zum Schreiben unlesbarer Programme</b>		<b>589</b>
Lügen Sie in den Kommentaren	589	

## ***Inhaltsverzeichnis***

Verwenden Sie möglichst kurze Variablennamen	589
Nutzen Sie Copy&Paste ausgiebig	590
Seien Sie kreativ im Umgang mit dem Thesaurus	590
Legen Sie sich niemals fest	590
Wenn's mit dem Englisch hapert	591
Ziehen Sie Schreibfehler konsequent durch	591
Seien Sie modern und geben Sie Ihren Quellcode frei	591
Finden Sie Workarounds für eigene Fehler	592
Verzichten Sie auf lesbare Codeformatierungen	592
<b>Kapitel 19</b>	
<b>Zehn nützliche Internetadressen zu C</b>	<b>595</b>
comp.lang.c Frequently Asked Questions	595
A Programmer's Heaven	595
Codeguru.com	595
Planet Sourcecode	595
Das Nachschlagewerk zur Bibliothek	596
Noch ein Nachschlagewerk	596
Der C-Standard	596
Code::Blocks	596
Visual Studio Express Editions	596
c++.de	597
<b>Kapitel 20</b>	
<b>In zehn Schritten zum ersten Compiler</b>	<b>599</b>
Der Download	599
Code::Blocks installieren	599
Die Compileroptionen	600
Ihr erstes Programm	601
Den C-Code eingeben	602
Ihr Programm erstellen	603
Fehler beseitigen	604
Ihr Programm ausführen	605
Was noch zu sagen wäre	605
Ein Wort zu Code::Blocks und Public Domain	606
<b>Stichwortverzeichnis</b>	<b>607</b>