



mitp

Peter
Prinz

2. Auflage

C Das Übungsbuch

Testfragen und Aufgaben mit Lösungen

Inhaltsverzeichnis

	Einleitung	9
1	Grundlagen	11
	Verständnisfragen	12
	Aufgaben	14
	Lösungen zu den Verständnisfragen	17
	Lösungen zu den Aufgaben	18
2	Elementare Datentypen, Konstanten und Variablen	21
	Verständnisfragen	22
	Aufgaben	24
	Lösungen zu den Verständnisfragen	28
	Lösungen zu den Aufgaben	28
3	Verwendung von Funktionen	35
	Verständnisfragen	36
	Aufgaben	38
	Lösungen zu den Verständnisfragen	43
	Lösungen zu den Aufgaben	44
4	Formatierte Ein- und Ausgabe	49
	Verständnisfragen	50
	Aufgaben	53
	Lösungen zu den Verständnisfragen	56
	Lösungen zu den Aufgaben	57
5	Operatoren	63
	Verständnisfragen	64
	Aufgaben	67
	Lösungen zu den Verständnisfragen	71
	Lösungen zu den Aufgaben	71

6	Kontrollstrukturen	77
	Verständnisfragen	78
	Aufgaben	81
	Lösungen zu den Verständnisfragen	85
	Lösungen zu den Aufgaben	86
7	Symbolische Konstanten und Makros	93
	Verständnisfragen	94
	Aufgaben	97
	Lösungen zu den Verständnisfragen	100
	Lösungen zu den Aufgaben	101
8	Typumwandlungen	109
	Verständnisfragen	110
	Aufgaben	113
	Lösungen zu den Verständnisfragen	116
	Lösungen zu den Aufgaben	117
9	Vektoren und Strings	121
	Verständnisfragen	122
	Aufgaben	126
	Lösungen zu den Verständnisfragen	129
	Lösungen zu den Aufgaben	130
10	Funktionen	141
	Verständnisfragen	142
	Aufgaben	144
	Lösungen zu den Verständnisfragen	148
	Lösungen zu den Aufgaben	148
11	Speicherklassen	159
	Verständnisfragen	160
	Aufgaben	162
	Lösungen zu den Verständnisfragen	166
	Lösungen zu den Aufgaben	167
12	Bitoperatoren	177
	Verständnisfragen	178
	Aufgaben	180
	Lösungen zu den Verständnisfragen	185
	Lösungen zu den Aufgaben	186

13	Zeiger	201
	Verständnisfragen	202
	Aufgaben	204
	Lösungen zu den Verständnisfragen	209
	Lösungen zu den Aufgaben	210
14	Dynamische Speicherplatzverwaltung	221
	Verständnisfragen	222
	Aufgaben	225
	Lösungen zu den Verständnisfragen	230
	Lösungen zu den Aufgaben	231
15	Strukturierte Datentypen	247
	Verständnisfragen	248
	Aufgaben	250
	Lösungen zu den Verständnisfragen	260
	Lösungen zu den Aufgaben	261
16	High-Level-Dateizugriff	281
	Verständnisfragen	282
	Aufgaben	284
	Lösungen zu den Verständnisfragen	290
	Lösungen zu den Aufgaben	291
	Stichwortverzeichnis	313

Einleitung

Dieses Buch wendet sich an Leser, die ihre C-Kenntnisse durch »Learning by Doing« vertiefen möchten. Es ist ideal, um sich im Stil eines Workshops auf Prüfungen oder auf die Mitarbeit in einem C-Projekt vorzubereiten.

Die Gliederung des Stoffs entspricht der eines C-Lernbuches, beginnt also mit den Grundlagen und endet mit den komplexeren Themen. Aber es ist nicht wesentlich, wie Sie C gelernt haben. Jedes Kapitel beginnt mit einer Übersicht und Zusammenfassung des Stoffs, zu dem anschließend Fragen und Aufgaben gestellt werden. Beispielsweise ist das Thema des 12. Kapitels »Bitoperatoren«. Wenn Ihnen dann der Inhalt der Zusammenfassung zu Bitoperatoren vertraut ist, sollten Sie auch ohne größere Probleme die anschließenden Fragen und Aufgaben lösen können.

Jedes Kapitel besteht neben der einführenden Beschreibung des Themas aus drei weiteren Teilen:

- Verständnisfragen
- Programmieraufgaben
- Musterlösungen zu allen Fragen und Aufgaben

Mit jeweils 20 Verständnisfragen können Sie testen, wie gut Sie sich in dem jeweiligen Themenbereich auskennen. Die Art der Fragen sind entweder Ja-Nein-Fragen, Multiple-Choice-Fragen oder es muss eine Aussage vervollständigt werden.

Im Aufgabenteil können Sie dann Ihr Wissen praktisch umsetzen. Dabei können Sie Ihren vertrauten C/C++-Compiler verwenden. In jedem Kapitel gibt es zehn Aufgaben mit steigendem Schwierigkeitsgrad. Die Bearbeitung einfacher Aufgaben ist oft in wenigen Minuten erledigt. Dagegen kann die Lösung umfänglicherer Aufgaben auch erheblich länger dauern. Dies gilt insbesondere bei Aufgaben zu den Themen »Dynamische Speicherplatzverwaltung«, »Strukturierte Datentypen« und »High-Level-Dateizugriff«. Umfangreichere Problemstellungen sind dabei oft auf mehrere Aufgaben verteilt.

Bei der Auswahl der Problemstellungen für Aufgaben wurde stets darauf geachtet, dass sie typisch und praxisnah sind. Auf diese Weise lernen Sie viele interessante Algorithmen und Datenstrukturen kennen. Auch durch die Verwendung vieler Standardfunktionen vertiefen Sie Ihre Kenntnisse über die Standardbibliothek. In jedem Fall verfügen Sie nach der Durcharbeitung des Buches über fundierte Programmiererfahrungen und einen umfangreichen Fundus von Beispiel-Code.

Trotz ausführlicher Aufgabenstellungen und vieler Hinweise kann es immer mal vorkommen, dass man nicht zum Ziel kommt. Dann hilft ein Blick in die kommentierten Musterlösungen. Außerdem ist es sicher immer interessant, die eigene Lösung mit der im Buch zu vergleichen. Die Musterlösungen zusammen mit Projekten für den C/C++ Compiler von Microsoft Visual Studio finden Sie auch im Internet unter <http://www.mitp.de/896>.

Ich bedanke mich bei allen, die an der Entstehung des Buches mitgewirkt haben, insbesondere bei Frau Schulz vom mitp-Verlag für die konstruktive Zusammenarbeit.

Dem Leser wünsche ich viele Erfolgserlebnisse beim Lösen der Übungen.

Peter Prinz

prinz_peter@t-online.de

Grundlagen

Dieses Kapitel enthält grundlegende Fragen und Aufgaben zur Erstellung von C-Programmen. Hierzu gehören folgende Themen:

- Header-Dateien inkludieren

Eine Header-Datei beinhaltet Informationen, die von einem C-Programm verwendet werden. Zum Beispiel enthält die Header-Datei `stdio.h` Informationen über die Funktionen der Standardbibliothek, die für die Ein-/Ausgabe von Daten zuständig sind. Eine Header-Datei wird mit der `#include`-Direktive in ein Programm kopiert.

- Anweisungen schreiben

Die Anweisungen eines Programms legen fest, was das Programm macht. Jede Anweisung schließt mit einem Semikolon ab. Zur Ausgabe von Daten auf den Bildschirm steht in C die Funktion `printf()` zur Verfügung. So gibt z.B. die Anweisung

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

den Text `Hallo!` aus und setzt den Cursor an den Anfang der nächsten Zeile.

- Eine `main`-Funktion definieren

Jedes C-Programm besteht im Wesentlichen aus Funktionen, die sich zur Laufzeit des Programms gegenseitig aufrufen. Die erste Funktion, die ausgeführt wird, ist stets die `main`-Funktion. Die auszuführenden Anweisungen stehen im Funktionsblock, d.h. innerhalb der Klammern `{ }`. Bei Erreichen einer `return`-Anweisung oder der abschließenden Klammer `}` des Funktionsblocks wird die Funktion verlassen. Die Ausführung einer `return`-Anweisung in der `main`-Funktion beendet also das Programm.

- Quelldateien kommentieren

Kommentare dienen zur Dokumentation des Quellcodes. Sie verbessern die Lesbarkeit und können bei der Fehlersuche nützlich sein. Jede Zeichenfolge, die in `/* ... */` eingeschlossen ist oder innerhalb einer Zeile mit `//` beginnt, ist ein Kommentar. Der Compiler ignoriert Kommentare.

Verständnisfragen

- 1.1 Ein C-Programm kann aus mehreren Quelldateien bestehen.
- ☐ Richtig
- ☐ Falsch
- 1.2 Die Übersetzung eines portablen C-Programms erzeugt eine ausführbare Datei, die auf allen Systemen lauffähig ist.
- ☐ Richtig
- ☐ Falsch
- 1.3 Eine Quelldatei wird mit einem _____ übersetzt.
- 1.4 Der _____ bindet eine Objektdaten mit anderen Modulen zu einer ausführbaren Datei.
- 1.5 Standardisierte Funktionen sind in der _____ enthalten.
- 1.6 Die Standardbibliothek ist eine Sammlung von
- a) C-Quelldateien.
- b) Objektdateien und Header-Dateien.
- c) ausführbaren Dateien.
- 1.7 Bei der Suche nach Fehlern in einem C-Programm beginnen Sie immer mit
- a) dem letzten vom Compiler angezeigten Fehler.
- b) irgendeinem angezeigten Fehler.
- c) dem ersten angezeigten Fehler.
- 1.8 Eine Warnung des Compilers kann einen
- a) Syntaxfehler anzeigen.
- b) logischen Fehler anzeigen.
- c) möglichen Laufzeitfehler anzeigen.
- 1.9 Jedes C-Programm enthält die Funktion _____.
- 1.10 In einem C-Programm bedeutet das Doppelkreuz # am Anfang einer Zeile, dass diese Zeile für
- a) den Compiler bestimmt ist.
- b) den Präprozessor bestimmt ist.
- c) die Header-Datei bestimmt ist.

- 1.11 Die Deklarationen der Standardfunktionen für die Ein-/Ausgabe befinden sich in der Header-Datei _____.
- 1.12 Die Programmausführung beginnt (abgesehen von allgemeinen Initialisierungen) mit
- a) der ersten `#include`-Direktive.
 - b) der ersten Anweisung in der Funktion `main()`.
 - c) der zuerst definierten Funktion.
- 1.13 Ein C-Programm gibt mit der `printf`-Funktion eine Meldung aus. Es enthält auch die Zeile

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

- a) weil jedes C-Programm diese Zeile enthalten muss.
 - b) zur Deklaration der Funktion `main()`.
 - c) zur Deklaration der Funktion `printf()`.
- 1.14 In der Funktion `main()` bewirkt die Anweisung
- ```
return 0;
```
- a) das Verlassen von `main()`.
  - b) die Beendigung des Programms.
  - c) die Rückgabe des Exitcodes 0 an das aufrufende Programm.

- 1.15 Die kürzeste Anweisung besteht aus \_\_\_\_\_.
- 1.16 C-Funktionen müssen in einer bestimmten Reihenfolge definiert werden.
- ☐ Richtig
  - ☐ Falsch
- 1.17 Die erste Funktion, die in einer Quelldatei definiert wird, ist stets die Funktion `main()`.
- ☐ Richtig
  - ☐ Falsch
- 1.18 Der Prototyp einer Funktion liefert dem Compiler die notwendigen Informationen, um die Funktion vor ihrer Definition aufzurufen.
- ☐ Richtig
  - ☐ Falsch

- 1.19 Zeichenfolgen werden als Kommentare interpretiert, wenn sie
- a) mit /\* beginnen.
  - b) in /\* \*/ eingeschlossen sind.
  - c) mit // beginnen.
- 1.20 In einer Zeile können mehrere Präprozessor-Direktiven stehen.
- ☐ Richtig
  - ☐ Falsch

## Aufgaben

- 1.1 Was gibt das folgende Programm auf dem Bildschirm aus?

```
#include <stdio.h>

int main()
{
 printf("Hi,\nWer geht");
 printf(" mit in den Biergarten");
 printf("?\n");
 return 0;
}
```

- 1.2 Formulieren Sie die entsprechenden Anweisungen, um

```
Los geht's!
```

- a) beginnend bei der aktuellen Cursorposition auszugeben.
  - b) am Anfang der nächsten Zeile auszugeben.
- 1.3 Jedes der folgenden Programme enthält zwei Fehler. Bestimmen und korrigieren Sie jeden Fehler.
- a)

```
#include <stdio>
int main()
{ // Eines von Murphy's Gesetzen
 print("Was schiefgehen kann, geht schief!\n");
 return 0;
}
```

b)

```
#include <stdio.h>
int main()
{
 printf("Was schiefgehen kann, geht schief!"\n)
}
```

c)

```
#include <stdio.h>
int main()
{
 / Wer hat das gesagt? /
 printf "Was schiefgehen kann, geht schief!\n";
 return 0;
}
```

- 1.4 Schreiben Sie ein C-Programm, das Ihren Namen, Ihre Telefonnummer und E-Mail-Adresse in je einer Zeile auf dem Bildschirm ausgibt.
- 1.5 Fügen Sie Kommentare in die Lösung zur Aufgabe 1.4 ein, und zwar einen Programmnamen, den Namen des Programmierers sowie eine Beschreibung, was das Programm macht.
- 1.6 Schreiben Sie ein C-Programm, das folgendes Menü ausgibt:

```
***** Meine Musiktitel *****

N = Neuen Eintrag hinzufügen
L = Eintrag löschen
F = Musiktitel finden
A = Alle Einträge anzeigen
B = Programm beenden

Ihre Wahl:
```

- 1.7 Sind die folgenden C-Programme vollständig und fehlerfrei?

a)

```
int main()
{
 return 0;
}
```

b)

```
include <stdio.h>
int main()
{
 return 0;
 printf "Da stimmt einiges nicht!\n";
}
```

c)

```
#include <stdio.h>
int main (
){ printf
("Alles klar?"); return
0;}
```

- 1.8 Angenommen, die folgenden Anweisungen befinden sich innerhalb einer main-Funktion. Was ist falsch?
- a) printf(Geben Sie eine Zahl ein: );
  - b) return "Alles klar!";
  - c) void pause() { printf("PAUSE!!!"); }
- 1.9 Verfolgen Sie den Ablauf des folgenden C-Programms und beschreiben Sie, was auf dem Bildschirm ausgegeben wird.

```
#include <stdio.h>
void stars2(void), stars6(void), stars10(void);

int main()
{
 stars2();
 stars6();
 stars10();
 stars2();
 stars2();
 return 0;
}

void stars2() { printf(" **\n"); }

void stars6() { printf(" *****\n"); }

void stars10() { printf(" *****\n"); }
```

- 1.10 Ändern Sie die `main`-Funktion aus der letzten Aufgabe so, dass folgende Grafik ausgegeben wird:

```

 * *
 * *
 * *
 * *
 * *
 * *
 * *

```

Fügen Sie außerdem Kommentare in den Quellcode ein, die erklären, was das Programm und die einzelnen Anweisungen machen.

## Lösungen zu den Verständnisfragen

- 1.1 Richtig
- 1.2 Falsch (Das Programm muss für jedes System übersetzt werden.)
- 1.3 Compiler
- 1.4 Linker
- 1.5 Standardbibliothek
- 1.6 b)
- 1.7 c)
- 1.8 b) und c)
- 1.9 `main()`
- 1.10 b)
- 1.11 `stdio.h`
- 1.12 b)
- 1.13 c)
- 1.14 a), b) und c)
- 1.15 Einem Semikolon
- 1.16 Falsch
- 1.17 Falsch
- 1.18 Richtig
- 1.19 b) und c)
- 1.20 Falsch

## Lösungen zu den Aufgaben

I.1

```
Hi,
Wer geht mit in den Biergarten?
```

I.2 a)

```
printf("Los geht's!");
```

b)

```
printf("\nLos geht's!");
```

I.3

a) In der ersten Zeile muss `stdio.h` statt `stdio` stehen.Die Funktion zur Ausgabe heißt `printf()`, nicht `print()`.b) Das Steuerzeichen `\n` muss im String stehen und die Anweisung mit einem Semikolon enden. Richtig ist also

```
printf("Was schiefgehen kann, geht schief!\n");
```

*Hinweis:* Die Anweisung

```
return 0;
```

darf in der `main`-Funktion fehlen. Sie wird dann automatisch am Ende der Funktion eingefügt.c) Innerhalb der Funktion `main()` ist der Kommentar nicht korrekt. Richtig sind folgende Kommentare:

```
// Wer zum Teufel hat das gesagt?
/* Wer zum Teufel hat das gesagt? */
```

Außerdem muss das Argument der Funktion `printf()` wie in Teil a) oder b) in Klammern stehen.

I.4

```
#include <stdio.h>
int main()
{
 printf("Eva Sommer\n") ;
 printf("Tel. +49 /89/ 9182730\n");
 printf("eva.s@yahoo.com\n");
 return 0;
}
```

1.5

```
// -----
// Programmname: ex01_05.c
// Autor: Eva Sommer
// Das Programm gibt einen Namen, eine Tel.-Nr.
// und eine E-Mail-Adresse auf dem Bildschirm aus.
// -----
#include <stdio.h>
int main()
{
 // Wie in der Lösung zur Aufgabe 1.4.
}
```

1.6

```
// -----
// ex01_06.c
// Gibt ein Menü für ein Musikverzeichnis aus.
// -----
#include <stdio.h>
int main()
{
 printf("***** Meine Musiktitel *****\n\n");

 printf(" N = Neuen Eintrag einzufuegen\n");
 printf(" L = Eintrag loeschen\n");
 printf(" F = Musiktitel finden\n");
 printf(" A = Alle Eintraege anzeigen\n");
 printf(" B = Programm beenden\n\n");

 printf("Ihre Wahl: ");

 printf("\n\n");
 return 0;
}
```

- 1.7 a) Das Programm tut zwar nichts, der Quellcode ist aber fehlerfrei und vollständig.
- b) Im Quellcode gibt es drei Fehler:
1. Das Zeichen # fehlt vor `include`.
  2. Die Anweisung `return 0;` beendet die `main`-Funktion. Sie muss also hier am Ende der Funktion stehen.
  3. Das Argument von `printf()` muss in Klammern stehen.
- c) Der Quellcode ist fehlerfrei und vollständig, aber schlecht lesbar.

- 1.8 a) Bei dem String fehlen die Hochkommata. Richtig ist also:

```
printf("Geben Sie eine Zahl ein: ");
```

- b) Der Return-Wert der main-Funktion muss eine Ganzzahl sein.  
c) Die Definition der Funktion pause() ist korrekt. Sie darf aber nicht innerhalb einer Funktion stehen, auch nicht innerhalb von main().

1.9

```
 **

 **
 **
```

1.10

```
/* -----
 * ex01_10.c
 * Gibt das Muster aus Aufgabe 1.10 aus.
 * -----
 */
#include <stdio.h>

void stars2(void), // Prototypen der verwendeten
 stars6(void), // Funktionen.
 stars10(void);

int main()
{
 stars10(); // *****
 stars6(); // *****
 stars2(); // **
 stars6(); // *****
 stars10(); // *****
 return 0;
}

void stars2() // Gibt 4 Blanks und 2 Sterne aus.
{ printf(" **\n"); }

void stars6() // Gibt 2 Blanks und 6 Sterne aus.
{ printf(" *****\n"); }

void stars10() // Gibt 10 Sterne aus.
{ printf("*****\n"); }
```



# Stichwortverzeichnis

## Symbole

\_\_STDC\_LIB\_EXT1\_\_ 50

#define 93

#endif 96

#ifdef 93

#ifndef 93, 96

#include 11, 35

#undef 93

## A

ABS 98

addArr() 226

Adresse 201

Adressoperator 202

Algorithmus

    Merge- 228

Anweisung 11

    do-while 77

    for 77

    if else 77

    return 11, 142

    switch 77

    while 77

Argument 35, 36

    Adress- 49

Array 121

ASCII-Code 25

assert() 99

Aufruf

    Funktions- 35

Ausdruck 63

Ausgabe

    dezimal 49

    exponentiell 49, 53

    Feldbreite 49

    Fixpunkt 49, 53

    formatierte 49

    -Genauigkeit 49

    hexadezimal 49

    linksbündig 51

    oktal 49

    rechtsbündig 52

Auswahloperator ? 77

auto 159

## B

Bedingung 69, 77

Binärmodus 281

Binomialkoeffizient 147

binomialkoeffizient() 147

Bit

    gelöschtes 177

    gesetztes 177, 180

    invertieren 177, 179

    löschen 179

    -Maske 179

    setzen 179

Bitfeld 247, 259, 262

    Ausrichtung 250

    Breite 247, 262

Bitgruppe 180, 182

Bitmuster 128, 178, 182, 183,  
    260

Bitoperator 177

    Links-Shift 177

    logische 177

    Operanden 177

    Rechts-Shift 177

bitPattern() 146, 183

Block 77, 159

break 77

Bruch

    Nenner 254

    Zähler 254

Bruchrechnung 254

bruchTodouble() 255

Bubble Sort 183

bubbleSort() 183

Byte

    High- 181

    Low- 181

## C

calloc() 221

case-Marke 91

Cast-Operator 109

Celsius 68

celsius2fahr() 144

char 21

Code

    ASCII- 25, 49

    Zeichen- 28

Compiler 12

Compilierung

    bedingte 96

cone() 206

const 24

const char \* 207

continue 77

ctime() 286

ctype.h 93

## D

Datei 281

    aktuelle Position 281

    ausführbare 12, 141, 143

    Binärmodus 281

    -Ende 281

    -Flags 281

    Header- 11

    mischen 289

    öffnen 281

    open-for-update 282

    -Operationen 281

    -Puffer 281

    schließen 281

    suchen in 289

    Textmodus 281

    Zugriffsmodus 281

Dateizugriff 281

    wahlfreier 281

Datensatz 247

    Aufbau 248

Datentyp 21, 22

DBL\_MAX 26

DBL\_MIN 26

default 91

Definition

    Variablen- 25

Deklaration  
 extern- 159, 160, 163  
 Funktions- 35  
 globaler Vektor 163  
 Parameter- 35  
 Variablen- 24  
 Dezimal 21  
 Differenz 63  
 Direktive  
 #define 93  
 #ifdef 93  
 #ifndef 93  
 #include 11, 35  
 #undef 93  
 displayFile() 285  
 Division 63  
 do while 77  
 Dokumentation 11  
 double 21  
 Dreieck  
 gleichseitiges 68  
 Durchschnitt 127  
 gleitender 165  
 dynamisch  
 Speicher 221  
 Vektor 222

**E**

Einer-Kompliment 177  
 Eingabefeld 49  
 else-if-Kette 77, 80  
 endkapital() 145  
 Endlosschleife 78  
 EOF 281  
 Eratosthenes  
 Sieb des 129  
 Escape-Sequenz 21, 23  
 Exklusiv-ODER 177  
 exp() 41, 85  
 Exponentiell  
 Notation 21, 26, 49  
 extern 159  
 extern-Deklaration 159, 160,  
 163

**F**

fahr2celsius() 144  
 Fahrenheit 68  
 Fakultät 81  
 fclose() 281  
 Fehler  
 Compiler- 12

Fehlersuche 11  
 Feldbreite 49  
 feof() 281  
 ferror() 281  
 fflush() 291  
 fgetpos() 281  
 fgets() 281, 291  
 Fibonacci-Quotienten 83  
 Fibonacci-Zahl 164  
 Fibonacci-Zahlen 82  
 FILE 281  
 FILE-Pointer 281  
 Filterprogramm 93, 97, 99,  
 100, 184  
 find\_int() 208  
 Fixpunktzahl 49  
 float 21  
 floor() 39  
 FLT\_MANT\_DIG 259  
 FLT\_MAX 26  
 FLT\_MIN 26  
 fopen 281  
 for 77  
 Formatelement 24, 49  
 Formatstring 49  
 fprintf() 281, 284, 287  
 fputs() 281  
 fread() 281  
 free() 221  
 fscanf() 281, 287  
 fseek() 281, 288  
 fsetpos() 281  
 ftell() 281, 288  
 Füllzeichen 52  
 Funktion 141  
 Argument 142  
 Aufruf 35, 40  
 -Block 141  
 Definition 141  
 Deklaration 35  
 inline 141  
 -Kopf 141  
 main() 11, 141  
 Parameter 141  
 Prototyp 142  
 Reihenfolge 142  
 rekursive 141, 147  
 Return-Wert 141  
 secure- 50  
 Typ void 142  
 Funktionsmakro 93  
 Fußgesteuert 77  
 fwrite() 281, 291

**G**

Ganzzahlerweiterung 109,  
 177  
 Geltungsbereich 159, 160  
 Genauigkeit 21, 26, 49  
 Standardwert 49  
 Geräte 281  
 getBitfield() 182  
 getc() 281  
 getMovingAverage() 166  
 gets\_s() 131  
 gets() 131  
 ggT() 255  
 Gleitpunkttyp 21  
 Gleitpunktzahl 49  
 Exponent 259  
 Mantisse 259  
 Globale Variable 159  
 goto 77  
 Groß-/Kleinschreibung 21

**H**

Hanoi  
 Türme von 256  
 Header-Datei 11, 35  
 assert.h 99  
 ctype.h 93, 100  
 float.h 26, 259  
 limits.h 22, 26, 54  
 math.h 35  
 Standard- 35  
 stdbool.h 39  
 stdio.h 11, 282  
 stdlib.h 37, 40, 221  
 time.h 40, 251  
 Hexadezimal 21  
 Hochkomma 21

**I**

if else 77  
 Index 121  
 Initialisierung  
 Liste 121  
 String 121  
 Variablen- 21  
 Vektor 121  
 initMovingAverage() 165  
 InitRandom 98  
 initTOH() 257  
 Inkrementoperator 63  
 inline 141, 146  
 int 21

iProductP3D() 253  
 iscntrl() 100  
 isdigit() 93, 98  
 islower() 93  
 isReadable() 284  
 isspace() 93, 100  
 isupper() 93  
 isxdigit() 98

## K

Kegel  
   Mantelfläche 206  
   Volumen 206  
 Kommentar 11, 14, 15  
 Kompilierung  
   bedingte 93  
 Kongruenzmethode  
   lineare 162  
 Konstante 22, 24  
   dezimal 21  
   Gleitpunkt- 21  
   hexadezimale 21, 28  
   numerische 21  
   oktale 21, 29  
   String- 21  
   symbolische 93  
   Zeichen- 21  
 Konvertierungsspezifikation  
   on 49  
 Koordinaten  
   kartesische 253  
 Kreditraten 69  
 kuerzenB() 255

## L

Laufbedingung 77  
 Lebensdauer 160  
   von Objekten 159  
 Linker 159  
 Linksbündig 51  
 Links-Shift 177  
 Liste  
   einfach verkettete 255  
 Literal 21  
 localtime() 251  
 log() 41  
 Lokale Variable 159  
 long 21  
 long double 21  
 long long 21  
 lotto() 208  
 Lottospiel 128  
 Low Byte 203

## M

main() 11, 15  
 Makro 93  
   ABS 95  
   Aufruf 93  
   Definition 93  
   Funktions- 93  
   MAX 98  
   MIN 98  
   Neudefinition 95  
   Random 98  
   swap() 183  
 malloc() 221  
 Marke  
   case- 91  
   Sprung- 77  
 math.h 35  
 Matrix 121  
 MAX 98  
 Maximum 77, 209  
 Mehrfach  
   -Inkludierung 93  
 memory  
   allocation 221  
 Memory Leak 236  
 merge() 228  
 Merge-Algorithmus 228  
   extern 289  
 MIN 98  
 Minimum 209  
 Mischen  
   von Dateien 289  
 Mittel  
   arithmetisches 146  
   geometrisches 146  
   harmonisches 146  
 Mittelwert 209  
 Modulo-Division 63, 67  
 moveTOH() 258  
 moveTower() 258  
 mulPolynomial() 230  
 Multiplikation 63  
 Murphy 25, 288  
 myRandom() 163

## N

Name 21  
   Parameter- 36  
 Namen 23  
 Name-Wert-Paar 127  
 NDEBUEG 99  
 nextFibo() 164  
 nextMovingAverage() 166

NICHT 63, 177, 179  
 NULL 201  
 Null-Erweiterung 111, 115  
 Null-Zeiger 208, 221

## O

Objektdatei 12  
 ODER 63, 177  
 Oktal 21  
 Operand 63  
 Operator 63  
   Adress- 201  
   arithmetische 63  
   Auswahl- 77  
   binär 63  
   Bit- 177  
   Cast- 109  
   Inkrement- 63  
   logische 63  
   Pfeil- 247  
   Punkt- 247  
   Shift- 177  
   sizeof 22  
   unär 63  
   Vergleichs- 63  
   Verweis- 201  
   Vorzeichen- 63  
   Zuweisung 63

## P

Palindrom 226  
 Parameter 35, 142  
   Deklaration 141  
   Makro- 93  
   -Namen 35  
   -Typ 35  
   Zeiger 204  
 Pfadangabe 281  
 Pfeil-Operator 247  
 Pointer 201  
 Polynom 229  
   Koeffizienten 229  
   Multiplikation 230  
 popJQ() 256  
 Potenzreihe  
   Basis 16 99  
   Basis 2 84  
   exp(x) 84  
   Zahldarstellung 84,  
     99  
 pow() 37, 39, 147  
 power() 147  
 Präprozessor 93

Primzahlen 129  
 printf() 11, 49  
 printJQ() 256  
 printTOH() 257  
 Priorität 63  
 Produkt  
   inneres 253  
 Programmfluss 77  
 Prototyp 35, 38  
   Funktions- 13  
 Punkt  
   dreidimensionaler 237  
 Punkt-Operator 247, 249  
 pushJQ() 256  
 putchar() 180, 182  
 putc() 281

## Q

Quelldatei 12  
 Queue 255

## R

rand() 37  
 Random 98  
 readFileID() 287  
 readJQ() 287  
 realloc() 221  
 Rechtsbündig 52  
 Rechts-Shift 177  
 record 247  
 register 159, 165  
 Rekursion 258  
 rekursiv 141, 147  
 resetFibo() 164  
 restoreTOH() 290  
 return 11, 13, 142  
 Return-Wert 35  
 reverse\_d() 165  
 rewind() 281  
 rotateBits() 182  
 Round2() 116  
 Rundungsfehler 114

## S

saveTOH() 290  
 scanf\_s() 50  
 scanf() 49  
   Feldbreite 52  
   Return-Wert 52  
 Schaltjahr 69  
 Schleife 77, 78  
   do while 77  
   for 77

  fußgesteuert 77  
   while 77  
 Schleifenkopf 77  
 searchStr() 289  
 Selection Sort 128, 147  
 selectionSort() 147  
 Semikolon 11  
 setBitfield() 182  
 Shift  
   arithmetischer 177  
   Division 177  
   logischer 177  
   Multiplikation 177  
 short 21  
 signed 21  
 size\_t 221  
 sizeof 22, 26  
 Skalarprodukt 253  
 Sortieren durch Auswahl 128  
 Sparrate 145  
 Speicher  
   dynamischer 221  
   freigeben 221, 223  
   Größe ändern 221  
   Initialisierung 221  
   reservieren 221  
 Speicherbelegung  
   float-Variable 259  
 Speicherklasse 159  
   auto 159  
   extern 159  
   register 159  
   -Spezifizierer 159  
   static 159  
   von Funktionen 159  
   von Objekten 159  
 Speicherleck 236  
 Spiel  
   Türme von Hanoi 256  
 Spielbrett 184  
   clearBoard() 184  
   getField() 184  
   setField() 184  
 Spieler 254  
   Punktstand 254  
 sprintf() 254  
 sProductP3D() 253  
 Sprung  
   bedingungsfreier 77  
   -Marke 77  
   Unterprogramm- 141  
 sqrt() 35, 69  
 srand() 37, 40  
 Stack 159

## Standard

-Ausgabe 49  
 -Bibliothek 35  
 -Eingabe 49  
 -Funktion 35  
 -Header-Datei 35  
 -Makros 93  
 -Streams 281  
 Standardabweichung 209  
 Standardbibliothek 12  
 static 159  
 static-Funktion 159, 162  
 static-Variable  
   Initialisierung 161  
 statistik() 209  
 stdbool.h 39  
 stderr 281  
 stdin 49, 281  
 stdlib.h 37  
 stdout 49, 281  
 Steuerzeichen 99  
 strcat() 128  
 strcmp() 130, 207  
 strcpy\_s() 122  
 strcpy() 121, 124, 128  
 Stream 281  
 strcmp() 207  
 String 121  
   dynamischer 226  
   einfügen 146  
   -Endezeichen 28  
   ersetzen 228  
   Initialisierung 121  
   invertieren 226  
   Vergleich 207  
 Stringende 121  
 strInsert() 146  
 strlen() 121, 229  
 strncmp() 229  
 strReplace() 228  
 strReplaceAll() 229  
 strReverse() 226  
 strstr() 229, 289  
 struct 247  
 struct Bruch 267  
 struct Gambler 254  
 struct Job 256  
 struct JobQueue 256  
 struct Point3D 253  
 struct QueueEl 256  
 struct Time 247  
 struct tm 252  
 struct Tower 257

- Struktur 247
  - Definition 247
  - Elemente 247
  - Elementzugriff 247
  - FILE 281
  - Initialisierung 247
  - sizeof 248
  - Typ 247
  - Variable 247
  - Zeiger auf 247
  - Zuweisung 249
- substring() 226
- Summe 63
  - Vektor 226
- sumP3D() 253
- swap() 183
- swapBytes() 181
- switch 77
  
- T**
- tan() 43
- Teiler
  - größter gemeinsamer 255
- Teilstring
  - ersetzen 228
  - kopieren 226
- Temperatur
  - Celsius 68
  - Fahrenheit 68
- time\_t 252
- time.h 40, 251
- time() 40, 251
- tolower() 93
- Ton 25
- toStringP3D() 254
- toupper() 93
- Türme von Hanoi 256
  - einlesen 290
  - Menü 259
  - speichern 290
  - Spielstatus 257
- Turmhöhe 42
- Typ 21
  - Breite 110
  - time\_t 252
- typedef 184, 247, 256
- Typhierarchie 109, 110
- Typumwandlung
  - bei Funktionsaufrufen 109, 115
  - bei Zuweisungen 109
  - Bitmuster 111
  - Datenverlust 109
  - explizite 109
  - implizite 109
  - Rundungsfehler 112
  - übliche arithmetische 109
  - Warnung 112
  
- U**
- UINT\_MAX 54
- Umlenkung
  - Ein-/Ausgabe 93, 97
  - Kommandozeile 97
- UND 63, 177
- Union 247, 260
  - Anfangsadresse 247
  - Größe 250
- union 247
- unsigned 21
- Unterprogramm 141
  
- V**
- Variable 21
  - Adresse 201
  - globale 24, 159
  - lokale 24, 144, 159
- Vektor 121
  - Definition 121
  - dynamischer 222
  - eindimensionaler 121
  - globaler 163
  - Initialisierung 121
  - mehrdimensionaler 121
  - Summe 226
- Vektorelement
  - Adresse 124
  - Zeiger auf 201
- Vergleich
  - lexikografisch 207
- Vergleichsoperator 63
- Verzweigung 77
  - Auswahloperator 77
  - if else 77
  - switch 77
- void 35
- void-Zeiger 221
- Vorrang 63
  - Klammern 66
- Vorzeichen 21
- Vorzeichen-Erweiterung 111, 115
  
- W**
- Warnung
  - Compiler- 12
- Warteschlange 255, 286
- einlesen 286
- Element einfügen 256
- Element entnehmen 256
- mit Prioritäten 255
- speichern 286
- Wertebereich 21, 26
- while 77
- Wörter zählen 100
- writeFileID() 287
- writeJQ() 286
- Wurzel 69
  
- Z**
- Zahl
  - binäre 83
  - dezimale 83
- Zeichen
  - Klassifizierung 93
- Zeichencode 49, 56, 82
- Zeiger 201
  - als Parameter 204, 206, 209
  - Arithmetik 201, 203
  - Definition 201
  - Differenz 203
  - Initialisierung 202
  - konstanter 201
  - Objektgröße 201
  - Return-Wert 204, 208
  - Typ 201, 202
  - und Vektoren 207
  - Variable 201
  - Vergleich 201
  - void- 221
  - Zuweisung 202
- Zeigerversion 206
- Zeilen zählen 100
- Zeit
  - lokale 252
- Zeitstempel 286
- Zufallszahl 37, 41
- Zugriffsmodus 282
- Zuweisung
  - einfache 63
  - Mehrfach- 63
  - zusammengesetzte 63, 177
- Zwischenraumzeichen 23, 49
- Zylinder
  - Mantelfläche 67
  - Volumen 67