mitp

Peter
Prinz

2. Auflage

C Das Übungsbuch

Testfragen und Aufgaben mit Lösungen

Inhaltsverzeichnis

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 Objektdatei 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

1.1

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

1.2 **a)**

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

b)

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

1.3 **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.

1.4

```
#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