

Hardwarenahe Programmierung / Angewandte Informatik

Musterlösung zu den Übungsaufgaben – 10. Oktober 2016

Aufgabe 1: Multiplikationstabelle

Geben Sie mit Hilfe einer Schleife ein „Einmaleins“ aus.
Dabei sollen die Faktoren und Ergebnisse rechtsbündig untereinander stehen:

```
1 * 7 = 7
2 * 7 = 14
...
10 * 7 = 70
```

Hinweis: Verwenden Sie Formatspezifikationen wie z. B. `%3d`
(siehe dazu die Dokumentation zu `printf()`, z. B. `man 3 printf`)

Lösung

Drei verschiedene richtige Lösungen finden Sie in den Dateien [loesung-1-1.c](#), [loesung-1-2.c](#) und [loesung-1-3.c](#).

Zum Compilieren von [loesung-1-2.c](#) und [loesung-1-3.c](#) ist mindestens der C99-Standard erforderlich (in `gcc`: Option `-std=c99` mit angeben).

Die Lösung in [loesung-1-3.c](#) ist zwar richtig, aber unnötig kompliziert und daher nicht empfohlen.

Eine **falsche** Lösung finden Sie in der Datei [loesung-1-f4.c](#): In der Ausgabe dieses Programms stehen die Faktoren und Ergebnisse nicht rechtsbündig untereinander.

Aufgabe 2: Fibonacci-Zahlen

Die Folge der Fibonacci-Zahlen ist definiert durch:

- 1. Zahl: 0
- 2. Zahl: 1
- nächste Zahl = Summe der beiden vorherigen

Schreiben Sie ein Programm, das die ersten 50 Fibonacci-Zahlen ausgibt.

Lösung

Zwei verschiedene richtige Lösungen finden Sie in den Dateien [loesung-2-1.c](#) und [loesung-2-2.c](#).

Die Lösung in [loesung-2-2.c](#) speichert alle berechneten Zahlen in einem Array, die in [loesung-2-1.c](#) hingegen speichert immer nur maximal drei Zahlen gleichzeitig. Sofern nicht alle berechneten Zahlen später noch benötigt werden, ist daher [loesung-2-1.c](#) zu bevorzugen.

Wichtig in [loesung-2-1.c](#) ist, daß `f0 + f1` berechnet wird, *bevor* `f0` oder `f1` ein neuer Wert zugewiesen wird. Dies ist nur möglich, weil das Programm eine zusätzliche Variable (hier: `f2`) verwendet.

Aufgabe 3: Schaltjahr ermitteln

Schreiben Sie ein Programm, das eine Jahreszahl erfragt und ausgibt, ob es sich um ein Schaltjahr handelt.

- Wenn die Jahreszahl durch 4 teilbar ist, ist das Jahr zunächst einmal ein Schaltjahr.
- Ausnahme: Wenn die Jahreszahl durch 100 teilbar ist, ist das Jahr kein Schaltjahr.
- Ausnahme von der Ausnahme: Wenn die Jahreszahl durch 400 teilbar ist, ist das Jahr doch wieder ein Schaltjahr.

Lösung

Am einfachsten ist es, die Aufgabenstellung in geschachtelte **if**-Verzweigungen zu übersetzen. Im folgenden finden Sie eine Funktion `is_leap_year()`, der man das Jahr übergibt und die für Schaltjahre **1** zurückgibt und für Nicht-Schaltjahre **0**.

```
#include <stdio.h>

int is_leap_year (int year)
{
    int leap_year = 0;
    if (year % 4 == 0)
    {
        leap_year = 1;
        if (year % 100 == 0)
        {
            leap_year = 0;
            if (year % 400 == 0)
                leap_year = 1;
        }
    }
    return leap_year;
}
```

(In C steht **0** für den Wahrheitswert „falsch“ und jeder Wert ungleich **0** für den Wahrheitswert „wahr“; die Zeile `leap_year = 0` steht daher wörtlich und selbsterklärend für „ist kein Schaltjahr“.)

Unter Verwendung von **else** läßt sich dies verkürzen zu:

```
#include <stdio.h>

int is_leap_year (int year)
{
    if (year % 4 == 0)
    {
        if (year % 100 == 0)
        {
            if (year % 400 == 0)
                return 1;
            else
                return 0;
        }
        else
            return 1;
    }
    else
        return 0;
}
```

Eine andere Möglichkeit ist es, die Schaltjahr-Bedingung in eine Kette von „und“- und „oder“-Verknüpfungen (C-Operatoren `&&` und `||`) zu übersetzen:

```
int is_leap_year (int year)
{
    if (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0))
        return 1;
    else
        return 0;
}
```

Dies ist zwar kürzer, aber nicht unbedingt übersichtlicher. Der erzeugte Code ist übrigens *nicht* kürzer und/oder effizienter als bei der Verwendung mehrerer **if**-Verzweigungen. Wir empfehlen, daß Sie immer so programmieren, daß Sie selbst den maximalen Überblick über Ihr Programm behalten.

Ein Hauptprogramm, das die o. a. Funktion aufruft, könnte dann wie folgt aussehen:

```
int main (void)
{
    int year;
    printf ("Bitte_geben_Sie_eine_Jahreszahl_ein:_");
    scanf ("%d", &year);
    if (is_leap_year (year))
        printf ("Das_Jahr_%d_ist_ein_Schaltjahr.\n", year);
    else
        printf ("Das_Jahr_%d_ist_kein_Schaltjahr.\n", year);
    return 0;
}
```

In den Dateien [loesung-3-1.c](#) bis [loesung-3-3.c](#) finden Sie lauffähige Programme, die die o. a. Funktionen aufrufen. Beachten Sie, daß die Funktion *vor* dem Hauptprogramm deklariert werden muß, damit das Hauptprogramm sie kennt. (Es gibt Tricks, mit denen es auch anders geht, aber was hätten wir in diesem Zusammenhang davon?)

In [loesung-3-4.c](#) und [loesung-3-5.c](#) findet die Schaltjahr-Prüfung direkt im Hauptprogramm statt. Dies ist ebenfalls eine richtige Lösung der Aufgabe, schränkt aber die Wiederverwertbarkeit des Codes ein.

Die Datei [loesung-3-4.c](#) enthält darüberhinaus Codeverdopplungen, nämlich mehrere identische `printf()`-Aufrufe an unterschiedlichen Stellen. Dies ist schlechter Programmierstil („Cut-and-paste-Programmierung“).

Die besten Lösungen sind [loesung-3-2.c](#) und [loesung-3-3.c](#).

Zum Testen:

- 1900 ist kein Schaltjahr.
- 1902 ist kein Schaltjahr.
- 1904 ist ein Schaltjahr.
- 1996 ist ein Schaltjahr.
- 1998 ist kein Schaltjahr.
- 2000 ist ein Schaltjahr.
- 2002 ist kein Schaltjahr.
- 2004 ist ein Schaltjahr.
- 2014 ist kein Schaltjahr.
- 2015 ist kein Schaltjahr.
- 2016 ist ein Schaltjahr.

Hier noch ein Hinweis für Unix-Shell-Experten:

```
for y in 1 2 3 4 5; do
    clear
    for x in 1900 1902 1904 1996 1998 2000 2002 2004 2014 2015 2016; do
        echo $x | ./loesung-3-$y
    done
    sleep 2s
done
```