

Hardwarenahe Programmierung

Übungsaufgaben – 29. Oktober 2018

Aufgabe 1: Strings

Strings werden in der Programmiersprache C durch Zeiger auf **char**-Variable realisiert.

Wir betrachten die folgende Funktion (Datei: [aufgabe-1.c](#)):

```
int fun_1 (char *s1, char *s2)
{
    int result = 1;
    for (int i = 0; s1[i] && s2[i]; i++)
        if (s1[i] != s2[i])
            result = 0;
    return result;
}
```

- (a) Was bewirkt die Funktion?
- (b) Welchen Sinn hat die Bedingung „s1[i] && s2[i]“ in der **for**-Schleife?
- (c) Was würde sich ändern, wenn die Bedingung „s1[i] && s2[i]“ in der **for**-Schleife zu „s1[i]“ verkürzt würde?
- (d) Schreiben Sie eine eigene Funktion, die dieselbe Aufgabe erledigt wie **fun_1()**, nur effizienter.

Aufgabe 2: Fehlerhaftes Programm: Primzahlen

Das nebenstehende Primzahlprogramm (Datei: [aufgabe-2.c](#)) soll Zahlen ausgeben, die genau zwei Teiler haben, ist aber fehlerhaft.

Korrigieren Sie das Programm derart, daß ein Programm entsteht, welches alle Primzahlen kleiner 100 ausgibt.

```
#include <stdio.h>

int main (void)
{
    int n, i, divisors;
    for (n = 0; n < 100; n++)
        divisors = 0;
        for (i = 0; i < n; i++)
            if (n % i == 0)
                divisors++;
        if (divisors = 2)
            printf ("%d_ist_eine_Primzahl.\n", n);
    return 0;
}
```

Aufgabe 3: Datum-Bibliothek

Schreiben Sie eine Bibliothek (= Sammlung von Deklarationen und Funktionen) zur Behandlung von Datumsangaben.

Diese soll enthalten:

- einen **struct**-Datentyp **date**, der eine Datumsangabe speichert,
- eine Funktion **void date_print (date *d)**, die ein Datum ausgibt,
- eine Funktion **int date_set (date *d, int day, int month, int year)**, die ein Datum auf einen gegebenen Tag setzt und zurückgibt, ob es sich um ein gültiges Datum handelt (0 = nein, 1 = ja),
- eine Funktion **void date_next (date *d)**, die ein Datum auf den nächsten Tag vorrückt.

Schreiben Sie auch ein Programm, das die o. a. Funktionen testet.