

# Hardwarenahe Programmierung

## Übungsaufgaben – 31. Oktober 2019

### 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: Programm analysieren

Wir betrachten das folgende C-Programm (Datei: [aufgabe-2.c](#)):

```
char*f="char*f=%c%s%c;main(){printf(f,34,f,34,10);}";main(){printf(f,34,f,34,10);}
```

- (a) Was bewirkt dieses Programm?
- (b) Wofür stehen die Zahlen?
- (c) Ergänzen Sie das Programm derart, daß seine `main()`-Funktion `int main (void)` lautet und eine **return**-Anweisung hat, wobei die in Aufgabenteil (a) festgestellte Eigenschaft erhalten bleiben soll.

### Aufgabe 3: Fehlerhaftes Primzahl-Programm

Das nebenstehende Primzahlsuchprogramm (Datei: [aufgabe-3.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;
}
```