

Hardwarenahe Programmierung / Angewandte Informatik

Übungsaufgaben – 28. November 2016

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 90 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 15 Punkte (von insgesamt 31) erreichen.

Aufgabe 1: Länge von Strings

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

Beispiel: **char** *hello_world = "Hello,_world!\n"

Die Systembibliothek stellt eine Funktion **strlen()** zur Ermittlung der Länge von Strings zur Verfügung (**#include <string.h>**).

- (a) Auf welche Weise ist die Länge eines Strings gekennzeichnet? (1 Punkt)
- (b) Wie lang ist die Beispiel-String-Konstante "Hello,_world!\n", und wieviel Speicherplatz belegt sie? (2 Punkte)
- (c) Schreiben Sie eine eigene Funktion **int strlen (char *s)**, die die Länge eines Strings zurückgibt. (3 Punkte)

Wir betrachten nun die folgenden Funktionen (Datei: **aufgabe-1.c**):

```
int fun_1 (char *s)
{
    int x = 0;
    for (int i = 0; i < strlen (s); i++)
        x += s[i];
    return x;
}
```

```
int fun_2 (char *s)
{
    int i = 0, x = 0;
    int len = strlen (s);
    while (i < len)
        x += s[i++];
    return x;
}
```

- (d) Was bewirken die beiden Funktionen? (2 Punkte)
- (e) Von welcher Ordnung (Landau-Symbol) sind die beiden Funktionen hinsichtlich der Anzahl ihrer Zugriffe auf die Zeichen im String – und warum? Sie dürfen für **strlen()** Ihre eigene Version der Funktion voraussetzen. (3 Punkte)
- (f) Schreiben Sie eine eigene Funktion, die dieselbe Aufgabe erledigt wie **fun_2()**, nur effizienter. (4 Punkte)

Aufgabe 2: Fehlerhaftes Programm

Das nebenstehende Primzahlsuchprogramm (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. (5 Punkte)

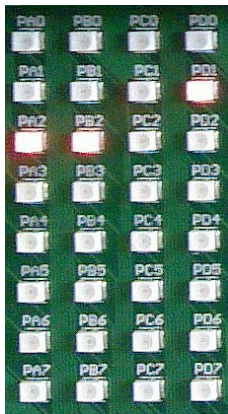
```
#include <stdio.h>

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: Mikro-Controller

An die vier Ports eines ATmega16-Mikro-Controllers sind Leuchtdioden angeschlossen:

- von links nach rechts
an die Ports A, B, C und D,
- von oben nach unten
an die Bits Nr. 0 bis 7.



Wir betrachten das folgende C-Programm (Datei: [aufgabe-3.c](#)) für diesen Mikro-Controller:

```
#include <avr/io.h>
#include <avr/interrupt.h>

int counter = 0;

ISR (TIMER0_COMP_vect)
{
    PORTA = 1 << ((counter++ >> 6) & 7);
}

int main (void)
{
    cli ();
    TCCR0 = (1 << CS01) | (1 << CS00);
    TIMSK = 1 << OCIE0;
    sei ();
    DDRA = 0xff;
    while (1);
    return 0;
}
```

Das Programm bewirkt ein periodisches Lauflicht in der linken Spalte von oben nach unten. Eine Animation davon finden Sie in der Datei [aufgabe-3.gif](#).

- (a) Wieso bewirkt das Programm überhaupt etwas, wenn doch das Hauptprogramm nach dem Initialisieren lediglich eine Endlosschleife ausführt, in der *nichts* passiert? (3 Punkte)
- (b) Erklären Sie, wie die Anweisung
- ```
PORTA = 1 << ((counter++ >> 6) & 7);
```
- das LED-Blinkmuster hervorruft. (6 Punkte)
- Hinweis: Zerlegen Sie die eine lange Anweisung in mehrere kürzere.  
Wenn nötig, verwenden Sie zusätzliche Variable für Zwischenergebnisse.
- (c) Was bedeutet „ISR (TIMER0\_COMP\_vect)“? (1 Punkt)
- (d) Wieso leuchten die Leuchtdioden PB2 und PD1? (2 Punkte)

*Viel Erfolg!*