

# Hardwarenahe Programmierung

## Übungsaufgaben – 12. Dezember 2022

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 100 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 17 Punkte (von insgesamt 34) erreichen.

### Aufgabe 1: Länge von Strings

Diese Aufgabe ist eine Neuauflage von Aufgabe 2 der Übung vom 7. November 2022, ergänzt um die Teilaufgaben (f) und (g).

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-2.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) Schreiben Sie eine eigene Funktion, die dieselbe Aufgabe erledigt wie **fun\_2()**, nur effizienter. (4 Punkte)
- (f) Von welcher Ordnung (Landau-Symbol) sind die beiden Funktionen hinsichtlich der Anzahl ihrer Zugriffe auf die Zeichen im String? Begründen Sie Ihre Antwort. Sie dürfen für **strlen()** Ihre eigene Version der Funktion voraussetzen. (3 Punkte)
- (g) Von welcher Ordnung (Landau-Symbol) ist Ihre effizientere Funktion? Begründen Sie Ihre Antwort. (1 Punkt)

## Aufgabe 2: Text-Grafik-Bibliothek

Schreiben Sie eine Bibliothek für „Text-Grafik“ mit folgenden Funktionen:

- **void clear (char c)**  
Bildschirm auf Zeichen `c` löschen,  
also komplett mit diesem Zeichen (z. B.: Leerzeichen) füllen
- **void put\_point (int x, int y, char c)**  
Punkt setzen (z. B. einen Stern (\*) an die Stelle  $(x, y)$  „malen“)
- **char get\_point (int x, int y)**  
Punkt lesen
- **void display (void)**  
das Gezeichnete auf dem Bildschirm ausgeben

Hinweise:

- Eine C-Bibliothek besteht aus (mindestens) einer `.h`-Datei und einer `.c`-Datei.
- Verwenden Sie ein Array als „Bildschirm“.  
Vor dem Aufruf der Funktion `display()` ist nichts zu sehen;  
alle Grafikoperationen erfolgen auf dem Array.
- Verwenden Sie Präprozessor-Konstante, z. B. `WIDTH` und `HEIGHT`,  
um Höhe und Breite des „Bildschirms“ festzulegen:  

```
#define WIDTH 72
#define HEIGHT 24
```
- Schreiben Sie zusätzlich ein Test-Programm, das alle Funktionen der Bibliothek benutzt,  
um ein hübsches Bild (z. B. ein stilisiertes Gesicht – „Smiley“) auszugeben.

(8 Punkte)

## Aufgabe 3: PBM-Grafik

Bei einer PBM-Grafikdatei handelt es sich um ein abgespeichertes C-Array von Bytes (`uint8_t`), das die Bildinformationen enthält:

- Die Datei beginnt mit der Kennung `P4`, danach folgen Breite und Höhe in Pixel als ASCII-Zahlen, danach ein Trennzeichen und die eigentlichen Bilddaten.
- Jedes Bit entspricht einem Pixel.
- Nullen stehen für Weiß, Einsen für Schwarz.
- MSB first.
- Jede Zeile des Bildes wird auf ganze Bytes aufgefüllt.

Viele Grafikprogramme können PBM-Dateien öffnen und bearbeiten. Der Anfang der Datei (Kennung, Breite und Höhe) ist auch in einem Texteditor lesbar.

Beispiel (`aufgabe-3.pbm`):

```
P4
14 14
<Bilddaten>
```



In dem untenstehenden Programmfragment (`aufgabe-3.c`) wird eine Grafik aus Textzeilen zusammengesetzt, so daß man mit einem Texteditor „malen“ kann:

```

#include <stdio.h>

/* ... */

int main (void)
{
    pbm_open (14, 14, "test.pbm");
    pbm_line ("                ");
    pbm_line ("      XXXXXX      ");
    pbm_line ("    X        X    ");
    pbm_line ("  _X        _X_ ");
    pbm_line (" _X        _X_ ");
    pbm_line ("X        X");
    pbm_line (" _X_XX_XX_ _X_ ");
    pbm_line (" _X_ _X_ _X_ ");
    pbm_line (" _X        _X_ ");
    pbm_line (" _X_X        _X_ ");
    pbm_line (" _X_ _X_ _X_ ");
    pbm_line (" _X_XXXX_ _X_ ");
    pbm_line ("    _X        ");
    pbm_line ("      XXXXXX      ");
    pbm_line ("                ");
    pbm_close ();
    return 0;
}

```

Ergänzen Sie das Programmfragment so, daß es eine Datei `test.pbm` erzeugt, die die Grafik enthält.

Das Programm soll typische Benutzerfehler abfangen (z. B. weniger Zeilen als in `pbm_open` angegeben), keine fehlerhaften Ausgaben produzieren oder abstürzen, sondern stattdessen sinnvolle Fehlermeldungen ausgeben.

Zum Vergleich liegt eine Datei `aufgabe-3.pbm` mit dem gewünschten Ergebnis bei, und die Datei `aufgabe-3.png` enthält dasselbe Bild.

(10 Punkte)

*Viel Erfolg!*