

# Hardwarenahe Programmierung

## Übungsaufgaben – 30. November 2023

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 75 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 13 Punkte (von insgesamt 26) erreichen.

### Aufgabe 1: PBM-Grafik

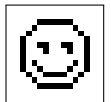
Bei einer PBM-Grafikdatei handelt es sich um ein abgespeichertes C-Array von Bytes (`uint8_t`), das die Bildinformationen enthält:

- 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-1.pbm`):

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



In dem untenstehenden Programmfragment (`aufgabe-1.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_ XX_ XX_ XX_ X_");
    pbm_line ("_X_ _X_ _X_ X_ X_");
    pbm_line (" _X_          _X_   ");
    pbm_line (" _X_X_          _X_X_");
    pbm_line (" _X_ _X_          _X_");
    pbm_line ("  _X_ XXXX_ _X_   ");
    pbm_line ("    _X_          _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-1.pbm](#) mit dem gewünschten Ergebnis bei, und die Datei [aufgabe-1.png](#) enthält dasselbe Bild.

(10 Punkte)

Hinweis für die Klausur: Abgabe in digitaler Form ist erwünscht, aber nicht zwingend.

## Aufgabe 2: Fakultät

Die Fakultät  $n!$  einer ganzen Zahl  $n \geq 0$  ist definiert als:

$$\begin{aligned} &1 \quad \text{für } n = 0, \\ &n \cdot (n - 1)! \quad \text{für } n > 0. \end{aligned}$$

Mit anderen Worten:  $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ .

Die folgende Funktion `fak()` berechnet die Fakultät *rekursiv* (Datei: [aufgabe-2.c](#)):

```
int fak (int n)
{
    if (n <= 0)
        return 1;
    else
        return n * fak (n - 1);
}
```

- (a) Schreiben Sie eine Funktion, die die Fakultät *iterativ* berechnet, d. h. mit Hilfe einer Schleife anstelle von Rekursion. (3 Punkte)
- (b) Wie viele Multiplikationen (Landau-Symbol) erfordern beide Versionen der Fakultätsfunktion in Abhängigkeit von  $n$ ? Begründen Sie Ihre Antwort. (2 Punkte)
- (c) Wieviel Speicherplatz (Landau-Symbol) erfordern beide Versionen der Fakultätsfunktion in Abhängigkeit von  $n$ ? Begründen Sie Ihre Antwort. (3 Punkte)

## Aufgabe 3: Einfügen in Strings (Ergänzung)

Diese Aufgabe ist eine Ergänzung von Aufgabe 3 der Übung vom 2. November 2023 um die Teilaufgaben (e), (f) und (g). Für den „Klausur-Modus“ können Sie die Teilaufgaben (a) bis (d) als „bereits gelöst“ voraussetzen.

Wir betrachten das folgende Programm ([aufgabe-1.c](#)):

```
#include <stdio.h>
#include <string.h>

void insert_into_string (char src, char *target, int pos)
{
    int len = strlen (target);
    for (int i = pos; i < len; i++)
        target[i+1] = target[i];
    target[pos] = src;
}

int main (void)
{
    char test[100] = "Hochshule_Bochum";
    insert_into_string ('c', test, 5);
    printf ("%s\n", test);
    return 0;
}
```

Die Ausgabe des Programms lautetet: `Hochschhhhhhhhhhhhh`

- (a) Erklären Sie, wie die Ausgabe zustandekommt.
- (b) Schreiben Sie die Funktion `insert_into_string()` so um, daß sie den Buchstaben `src` an der Stelle `pos` in den String `target` einfügt.

Die Ausgabe des Programms müßte dann `Hochschule Bochum` lauten.

- (c) Was kann passieren, wenn Sie die Zeile `char test[100] = "Hochshule_Bochum";` durch `char test[] = "Hochshule_Bochum";` ersetzen? Begründen Sie Ihre Antwort.
- (d) Was kann passieren, wenn Sie die Zeile `char test[100] = "Hochshule_Bochum";` durch `char *test = "Hochshule_Bochum";` ersetzen? Begründen Sie Ihre Antwort.
- (e) Schreiben Sie eine Funktion `void insert_into_string_sorted (char src, char *target)`, die voraussetzt, daß der String `target` alphabetisch sortiert ist und den Buchstaben `src` an der alphabetisch richtigen Stelle einfügt. Diese Funktion darf die bereits vorhandene Funktion `insert_into_string()` aufrufen. (4 Punkte)

Zum Testen eignen sich die folgenden Zeilen im Hauptprogramm:

```
char test[100] = "";
insert_into_string_sorted ('c', test);
insert_into_string_sorted ('a', test);
insert_into_string_sorted ('d', test);
insert_into_string_sorted ('b', test);
```

Danach sollte `test[]` die Zeichenfolge `"abcd"` enthalten.

- (f) Wie schnell (Landau-Symbol in Abhängigkeit von der Länge  $n$  des Strings) arbeitet Ihre Funktion `void insert_into_string_sorted (char src, char *target)`? Begründen Sie Ihre Antwort. (1 Punkt)
- (g) Beschreiben Sie – in Worten oder als C-Quelltext –, wie man die Funktion `void insert_into_string_sorted (char src, char *target)` so gestalten kann, daß sie in  $\mathcal{O}(\log n)$  arbeitet. (3 Punkte)

*Viel Erfolg!*