

# Hardwarenahe Programmierung

## Übungsaufgaben 10 – 9. Januar 2025

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 120 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 21 Punkte (von insgesamt 43) erreichen.

### Aufgabe 1: Personen-Datenbank

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

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

typedef struct
{
    char first_name[10];
    char family_name[20];
    char day, month;
    int year;
} person;

int main (void)
{
    person sls;
    sls.day = 26;
    sls.month = 7;
    sls.year = 1951;
    strcpy (sls.first_name, "Sabine");
    strcpy (sls.family_name, "Leutheusser-Schnarrenberger");
    printf ("%s_%s_wurde_am_%d.%d.%d_geboren.\n",
            sls.first_name, sls.family_name, sls.day, sls.month, sls.year);
    return 0;
}
```

Die Standard-Funktion `strcpy()` bewirkt ein Kopieren eines Strings von rechts nach links, hier also z. B. die Zuweisung der String-Konstanten "Sabine" an die String-Variable `sls.first_name[]`.

Das Programm wird für einen 32-Bit-Rechner compiliert und ausgeführt.

(Die `gcc`-Option `-m32` sorgt dafür, daß `gcc` Code für einen 32-Bit-Prozessor erzeugt.)

```
$ gcc -Wall -O -m32 aufgabe-2.c -o aufgabe-2
$ ./aufgabe-2
Sabine Leutheusser-Schnarrenberger wurde am 110.98.1701278309 geboren.
Speicherzugriffsfehler
```

- (a) Erklären Sie die Ausgabe des Programms einschließlich der Zahlenwerte. (4 Punkte)
- (b) Welche Endianness hat der verwendete Rechner? Begründen Sie Ihre Antwort. (1 Punkt)
- (c) Wie sähe die Ausgabe auf einem Rechner mit entgegengesetzter Endianness aus? (2 Punkte)
- (d) Erklären Sie den Speicherzugriffsfehler. (Es kann sein, daß sich der Fehler auf Ihrem Rechner nicht bemerkbar macht. Er ist aber trotzdem vorhanden.) (2 Punkte)

### Aufgabe 2: Einfügen in Strings (Ergänzung)

Diese Aufgabe ist eine Ergänzung von Aufgabe 3 der Übung 3 vom 7. November 2024 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-2.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 lautet: `Hochschhhhhhhhhhh`

- Erklären Sie, wie die Ausgabe zustandekommt.
- 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.
- Was kann passieren, wenn Sie die Zeile `char test[100] = "Hochshule_Bochum";` durch `char test[] = "Hochshule_Bochum";` ersetzen? Begründen Sie Ihre Antwort.
- Was kann passieren, wenn Sie die Zeile `char test[100] = "Hochshule_Bochum";` durch `char *test = "Hochshule_Bochum";` ersetzen? Begründen Sie Ihre Antwort.
- 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.

- 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)
- 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)

### Aufgabe 3: Objektorientierte Tier-Datenbank

Das unten dargestellte Programm (Datei: [aufgabe-3a.c](#)) soll Daten von Tieren verwalten.

Beim Compilieren erscheinen die folgende Fehlermeldungen:

```

$ gcc -std=c99 -Wall -O aufgabe-2a.c -o aufgabe-2a
aufgabe-2a.c: In function 'main':
aufgabe-2a.c:31: error: 'animal' has no member named 'wings'
aufgabe-2a.c:37: error: 'animal' has no member named 'legs'

```

Der Programmierer nimmt die in Rot dargestellten Ersetzungen vor  
(Datei: [aufgabe-3b.c](#)). Daraufhin gelingt das Compilieren, und die Ausgabe des Programms lautet:

```
$ gcc -std=c99 -Wall -O aufgabe-2b.c -o aufgabe-2b
$ ./aufgabe-2b
A duck has 2 legs.
Error in animal: cow
```

- (a) Erklären Sie die o. a. Compiler-Fehlermeldungen. (2 Punkte)
- (b) Wieso verschwinden die Fehlermeldungen nach den o. a. Ersetzungen? (3 Punkte)
- (c) Erklären Sie die Ausgabe des Programms. (5 Punkte)
- (d) Beschreiben Sie – in Worten und/oder als C-Quelltext – einen Weg, das Programm so zu berichtigen, daß es die Eingabedaten ("A duck has 2 wings. A cow has 4 legs.") korrekt speichert und ausgibt. (4 Punkte)
- (e) Schreiben Sie das Programm so um, daß es keine expliziten Typumwandlungen mehr benötigt. Hinweis: Verwenden Sie **union**. (4 Punkte)
- (f) Schreiben Sie das Programm weiter um, so daß es die Objektinstanzen **duck** und **cow** dynamisch erzeugt. Hinweis: Verwenden Sie **malloc()** und schreiben Sie Konstruktoren. (4 Punkte)
- (g) Schreiben Sie das Programm weiter um, so daß die Ausgabe nicht mehr direkt im Hauptprogramm erfolgt, sondern stattdessen eine virtuelle Methode **print()** aufgerufen wird. Hinweis: Verwenden Sie in den Objekten Zeiger auf Funktionen, und initialisieren Sie diese in den Konstruktoren. (4 Punkte)

```
#include <stdio.h>

#define ANIMAL 0
#define WITH_WINGS 1
#define WITH_LEGS 2

typedef struct animal
{
    int type;
    char *name;
} animal;

typedef struct with_wings
{
    int wings;
} with_wings;

typedef struct with_legs
{
    int legs;
} with_legs;

int main (void)
{
    animal *a[2];

    animal duck;
    a[0] = &duck;
    a[0]→type = WITH_WINGS;
    a[0]→name = "duck";
    a[0]→wings = 2; ← ((with_wings *) a[0])→wings = 2;

    animal cow;
    a[1] = &cow;
    a[1]→type = WITH_LEGS;
    a[1]→name = "cow";
    a[1]→legs = 4; ← ((with_legs *) a[1])→legs = 4;

    for (int i = 0; i < 2; i++)
        if (a[i]→type == WITH_LEGS)
            printf ("A_%s_has_%d_legs.\n", a[i]→name,
                    ((with_legs *) a[i])→legs);
        else if (a[i]→type == WITH_WINGS)
            printf ("A_%s_has_%d_wings.\n", a[i]→name,
                    ((with_wings *) a[i])→wings);
        else
            printf ("Error_in_animal:_%s\n", a[i]→name);

    return 0;
}
```

*Viel Erfolg!*