

Hardwarenahe Programmierung / Angewandte Informatik

Übungsaufgaben – 11. Dezember 2017

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 65 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 11 Punkte (von insgesamt 23) erreichen.

Aufgabe 1: Daten im Speicher

Das folgende C-Programm `aufgabe-1.c` gibt den Speicherbereich, in dem sich seine Variablen befinden, als eine Folge von 8-Bit-Zahlen aus:

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

int16_t a = -1;
int32_t b = 8320;

int main (void)
{
    uint8_t *p = &a;
    for (int i = 0; i < 8; i++)
        printf ("%d", p[i]);
    printf ("\n");
}
```

Das Programm wird ohne Optimierung auf einem 32-Bit-Rechner kompiliert (mit Warnung) und gestartet:

```
$ gcc -Wall aufgabe-1.c -o aufgabe-1
aufgabe-1.c: In function "main":
aufgabe-1.c:9:16: warning: initialization from incompatible pointer type [...]
$ ./aufgabe-1
255 255 0 0 128 32 0 0
```

- (a) Welche Endianness hat der verwendete Rechner und warum? (2 Punkte)
- (b) Erklären Sie die ausgegebenen Zahlen.
Zu welcher Variablen gehört jeweils die Zahl? (4 Punkte)
- (c) Wie würde die Ausgabe auf einem 16-Bit-Rechner mit entgegengesetzter Endianness lauten und warum? (3 Punkte)
- (x) **Freiwillige Zusatzaufgabe:** Erklären Sie, was sich ändert und warum, wenn das Programm auf einem 32-Bit-Rechner *mit* Optimierung (`-O`) kompiliert wird. (5 Extrapunkte)

Hinweis: Aus anderen Lehrveranstaltungen sollte Ihnen der Begriff des Zweierkomplements bekannt sein. Falls nicht, gilt eine Recherche nach diesem Begriff als zugelassenes Hilfsmittel.

Aufgabe 2: Zeigerarithmetik

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

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

void output (uint16_t *a)
{
    for (int i = 0; a[i]; i++)
        printf ("_%d", a[i]);
    printf ("\n");
}

int main (void)
{
    uint16_t prime_numbers[] = { 2, 3, 5, 7, 11, 13, 17, 0 };

    uint16_t *p1 = prime_numbers;
    output (p1);
    p1++;
    output (p1);

    char *p2 = prime_numbers;
    output (p2);
    p2++;
    output (p2);

    return 0;
}
```

Das Programm wird kompiliert und ausgeführt:

```
$ gcc -Wall aufgabe-2.c -o aufgabe-2
aufgabe-2.c: In function 'main':
aufgabe-2.c:20:13: warning: initialization from
                  incompatible pointer type [enabled by default]
aufgabe-2.c:21:3: warning: passing argument 1 of 'output' from
                  incompatible pointer type [enabled by default]
aufgabe-2.c:4:6: note: expected 'uint16_t *' but argument is of type 'char *'
aufgabe-2.c:23:3: warning: passing argument 1 of 'output' from
                  incompatible pointer type [enabled by default]
aufgabe-2.c:4:6: note: expected 'uint16_t *' but argument is of type 'char *'
$ ./aufgabe-2
 2 3 5 7 11 13 17
 3 5 7 11 13 17
 2 3 5 7 11 13 17
768 1280 1792 2816 3328 4352
```

- (a) Erklären Sie die Funktionsweise der Funktion `output ()`. (2 Punkte)
- (b) Begründen Sie den Unterschied zwischen der ersten (`2 3 5 7 11 13 17`) und der zweiten Zeile (`3 5 7 11 13 17`) der Ausgabe des Programms. (2 Punkte)
- (c) Erklären Sie die beim Compilieren auftretenden Warnungen und die dritte Zeile (`2 3 5 7 11 13 17`) der Ausgabe des Programms. (3 Punkte)
- (d) Erklären Sie die vierte Zeile (`768 1280 1792 2816 3328 4352`) der Ausgabe des Programms. Sie dürfen einen Little-Endian-Rechner voraussetzen. (4 Punkte)

Aufgabe 3: Text-Grafik-Bibliothek

Ergänzen Sie die Text-Grafik-Bibliothek aus der [Übung vom 6. 11. 2017](#) um eine weitere Funktion:

- **void fill (int x, int y, char c, char o)**
Fläche in der „Farbe“ `o`, die den Punkt `(x, y)` enthält, mit der „Farbe“ `c` ausmalen (3 Punkte)

Beispiel:

- `fill (3, 7, '+', ' ')`; soll ab den Koordinaten (3,7) eine zusammenhängende Fläche aus Leerzeichen (' ') suchen und alle diese Leerzeichen durch Plus-Zeichen ('+') ersetzen.
- Die zusammenhängende Fläche ist normalerweise mit anderen Zeichen umrandet.
Weitere Leerzeichen außerhalb dieser Umrandung sollen unverändert bleiben.

Hinweise:

- Führen Sie eine Web-Recherche nach dem Begriff „Floodfill“ durch.
- Schreiben Sie ein Test-Programm, das interessante umrandete Flächen – auch mit „Loch“ – „ausmalt“.

Viel Erfolg!