

## Angewandte Informatik – Klausur – 7. Februar 2013

Prof. Dr. Peter Gerwinski, Wintersemester 2012/13

Name:	
Matrikel-Nr.:	

Zeit: 120 Minuten

Zulässige Hilfsmittel:

- Schreibgerät
- Beliebige Unterlagen in Papierform und/oder auf Datenträgern
- Elektronische Rechner (Notebook, Taschenrechner o. ä.)  
*ohne* Zugang zu Datennetzen jeglicher Art

Nur die o. a. zulässigen Hilfsmittel dürfen sich während der Klausur im Arbeitsbereich befinden.  
WLAN-, Bluetooth- und sonstige Funkeinheiten von Notebooks o. ä. sind per Hardware auszuschalten.  
Mobiltelefone, Geräte mit mobilem Internet-Zugang u. ä. sind auszuschalten und in der Tasche zu verstauen.

Die reguläre Maximalpunktzahl beträgt 42 Punkte.  
Bei besonderen Leistungen sind Zusatzpunkte möglich.  
Mit 20 erreichten Punkten gilt die Klausur als bestanden.

## 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. (4 Punkte)

Wir betrachten nun die folgenden Funktionen:

```
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? (3 Punkte)

(f) Schreiben Sie eine eigene Funktion, die dieselbe Aufgabe erledigt wie `fun_2()`, nur effizienter. (4 Punkte)

## Aufgabe 2: Trickprogrammierung

Wir betrachten das folgende Programm [hallo.c](#):

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

int main(void)
{
    uint64_t x = 4262939000843297096;
    char *s = &x;
    printf ("%s\n", s);
    return 0;
}
```

Das Programm wird kompiliert und auf einem 64-Bit-Little-Endian-Computer ausgeführt:

```
$ gcc -Wall -O hallo.c -o hallo
hallo.c: In function 'main':
hallo.c:7:13: warning: initialization from incompatible pointer type [...]
$ ./hallo
Hallo
```

(a) Erklären Sie die Warnung beim Compilieren. (2 Punkte)

(b) Erklären Sie die Ausgabe des Programms. (4 Punkte)

- (c) Wie würde die Ausgabe des Programms auf einem 64-Bit-Big-Endian-Computer lauten? (5 Punkte)  
Hinweis: Es kann hilfreich sein, für die Umrechnung ein Programm zu schreiben.

### Aufgabe 3: Fehlerhaftes Programm

Das folgende Primzahlsuchprogramm soll Zahlen ausgeben, die genau zwei Teiler haben, ist aber fehlerhaft.

```
#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;
}
```

Korrigieren Sie das Programm derart, daß ein Programm entsteht, welches alle Primzahlen kleiner 100 ausgibt. (7 Punkte)

### Aufgabe 4: Prüfwert für Nachrichten

Bei Datenübertragung wird häufig zusätzlich zur eigentlichen Nachricht ein Prüfwert mitgesendet, um Übertragungsfehler erkennen (und evtl. sogar korrigieren) zu können. Ein häufig verwendeter Bestandteil dieses Prüfwerts ist die Anzahl der 1-Bits in der Nachricht.

- (a) Wieviele 1-Bits enthält die Nachricht "Hello, world!\n"? (1 Punkt)

[illegible]

- (b) Schreiben Sie eine Funktion `int count_bits (char *s)`, die für eine gegebene Nachricht `s` die Anzahl der 1-Bits zurückliefert. (7 Punkte)