Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

**TU Kaiserslautern**
**Fachbereich Informatik**

# Lösungshinweise zum Übungsblatt 9: Programmieren in C (WS 2018/19)

1. Dieses Übungsblatt ist ein **Pflichtübungsblatt**, d.h. Sie müssen bei der Bearbeitung dieses Blattes **mind. 50% der Gesamtpunkte** erreichen.
2. Die Abgabe erfolgt über das Exclaim-System gemeinsam mit Ihrem Teampartner.
3. Zur Beantwortung von Fragen und Hilfestellung bei der Bearbeitung kommen Sie bitte in die wöchentliche Fragestunde, Mittwoch 15:30, in Terminalraum 32-410.
4. Bitte laden Sie einzelne Dateien hoch, wie in der Aufgabenstellung angegeben, keine Archive, keine kompilierten Dateien.
5. Programme, die nicht kompilieren, werden nicht korrigiert und mit **0 Punkten** bewertet.
6. Bitte beachten Sie unsere Regelung bei Plagiaten:
   - Wenn Sie sekundäre Quellen, wie Bücher oder das Internet verwenden, müssen Sie immer die Quelle angeben. Das einfache Kopieren aus anderen Quellen ist für die Übungen nicht gestattet. Wenn wir in einer Übungsabgabe kopierten Code ohne Quellenangabe finden, wird die gesamte Abgabe mit 0 Punkten bewertet.
   - Sie können Übungsaufgaben gerne mit den Mitgliedern anderer Teams diskutieren. Sie sollten jedoch Ihren Code vor der Abgabefrist nicht an andere Teams weitergeben bzw. zeigen.
   - Wenn Code von anderen Teams kopiert wurde, werden die Abgaben **von beiden Teams** mit 0 Punkten bewertet.
   - Wir behalten uns vor Punkte auch nachträglich abzuziehen, wenn ein Verstoß erst später bemerkt wird.

## Rettet das Weihnachtsfest!

Bei der Auftaktbesprechung der diesjährigen Weihnachtssaison verkündet der Weihnachtsmann: "Liebe Weihnachtswichtel, Engel und Elfen! Wir werden dieses Jahr die Planung der Geschenke und Verarbeitung der Listen mittels neuester Technologie angehen. Wir steigen um von der Elfen-verarbeitetenden Datenverarbeitung auf die Elektronische Datenverarbeitung!"

Ihre Aufgabe ist es, dem Weihnachtsmann dabei zu unterstützen und Programme zu schreiben, die die Aufgaben der Wichtel, Engel und Elfen automatisieren.

## Aufgabe 1: Durchsuchen der Wunschzettel (10 Punkte)

Die Weihnachtswichtel haben von allen Menschen die Wunschzettel eingesammelt und eingescannt. Um die Produktion der Geschenke besser planen zu können, will der Weihnachtsmann wissen, auf welchem Wunschzettel ein bestimmtes Geschenk gewünscht wurde.

Ziel dieser Aufgabe ist es, ein C-Programm zu schreiben, das einen gegebenen Suchtext in einem Eingabetext finden kann und alle Zeilen des Eingabetexts ausgibt, welche den Suchtext enthalten.

a) Laden Sie von der Vorlesungshomepage die Vorlage `wunschzettel.c` herunter und ergänzen Sie diese wie in den nächsten Schritten beschrieben.

b) Schreiben Sie eine Funktion:

```
bool contains(char *search, int searchSize, char *input, int inputSize).
```

Die Funktion nimmt einen Suchtext `search` der Länge `searchSize` und einen Eingabetext `input` der Länge `inputSize`. Die Funktion soll `true` zurückgeben, wenn der gesuchte Text im Eingabe-Text vorkommt und ansonsten `false`.

Verwenden Sie **keine** Funktionen aus der C-Bibliothek für diese Aufgabe.

```c
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>

#define N 1024

bool contains(char *search, int searchSize, char *input, int inputSize)
{
    for (int i=0; i<inputSize-searchSize; i++) {
        bool found = true;
        for (int j=0; j<searchSize; j++)
        {
            if (input[i+j] != search[j])
            {
                found = false;
                break;
            }
        }
        if (found)
        {
            return true;
        }
    }
    return false;
}
```

c) Schreiben Sie eine `main`-Funktion für Ihr Programm, welches einen Suchtext als Programm-Parameter nimmt und die Standard-Eingabe nach Vorkommen des Suchstrings durchsucht. Die Zeilen der Eingabe, welche den Suchtext enthalten, sollen zusammen mit ihrer Zeilennummer ausgegeben werden. Dabei soll zuerst die Zeilennummer ausgegeben werden, dann ein Leerzeichen und dann der Text der Zeile.

Für das zeilenweise Lesen der Eingabe (Wunschzettel) verwenden Sie die Funktion `char *fgets(char *s, int n, FILE *stream)`. Diese Funktion nimmt einen Buffer `s` der Größe `n` und einen Eingabe-Strom. Wenn Sie `stdin` als Eingabe-Strom verwenden, wird von der Standardeingabe gelesen. Die Funktion liest bis eine neue Zeile startet, das Ende des Stroms erreicht ist oder die maximale Länge `n-1` erreicht ist. Rückgabe der Funktion ist `s`, falls das Einlesen erfolgreich war und `NULL` falls das Ende des Stroms erreicht wurde oder ein Fehler auftritt. Zeilenumbrüche am Ende der Zeile werden in `s` beibehalten. Der eingelesene String endet jeweils mit einem `'\0'`-Zeichen. Verwenden Sie zum Einlesen einen Buffer der Größe `1024`. Längere Zeilen müssen nicht korrekt behandelt werden, sollten aber nicht zu undefinierten Verhalten führen.

*Beispiel*
Programmparameter: `Schoko`

| *Eingabe:* | *Ausgabe:* |
|---|---|
| `Apfel, Nuss und Mandeln`<br>`Autoparkgarage und Schoko`<br>`Feuerwehrauto`<br>`Schokoladenlinsen`<br>`Filzpantoffel` | `2 Autoparkgarage und Schoko`<br>`4 Schokoladenlinsen` |

```c
int main(int argc, char **argv)
{
    char* searchString = argv[1];
    int searchStringLen = strlen(searchString);

    char buffer[N] = {0};
    int lineNr = 1;
    while (true)
    {
        char *s = fgets(buffer, N, stdin);
```

```
        if (s == NULL)
        {
            break;
        }
        if (contains(searchString, searchStringLen, s, strlen(s))) {
            printf("%d %s", lineNr, buffer);
        }
        lineNr++;
    }
}
```

## Aufgabe 2: Die fleißigsten Studierenden (10 Punkte)

Die Engel haben das Jahr Aufzeichnungen angefertigt, wer brav war und eine Liste aller Studierenden ermittelt, die regelmäßig an den Übungen teilgenommen haben. Um die Auszeichnung "Fleißigster Studi 2018" zu vergeben, benötigt der Weihnachtsmann für jeden Studiengang die Person mit der höchsten Punktzahl auf den gesamten Übungsblättern.

Ihre Aufgabe ist es ein Programm zu schreiben, das aus der Liste der Engel den fleißigsten Studierenden ermittelt und ausgibt.

a) Laden Sie von der Vorlesungshomepage die Vorlage `studis.c` herunter und ergänzen Sie diese wie in den nächsten Schritten beschrieben.

b) Modellieren Sie einen Studierenden als einen `struct Studi` mit folgenden Strukturvariablen:

- Name mit max. 20 Zeichen als String (`name`)
- Punktezahl als Integer (`punkte`)
- Studiengang als Wert eines `enum Fach {EIT, INF, MATH}` (`studiengang`)

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <limits.h>
#define N 61
#define N_STUDIS 100

enum Fach
{
    EIT,
    INF,
    MATH
};

struct Studi
{
    char name[21];
    int punkte;
    enum Fach studiengang;
};
```

c) In der Vorlage finden Sie eine Funktion `void init(struct Studi *s, char *eingabe)` , die die Informationen aus dem String `eingabe` nimmt und in die Struktur `s` einträgt. Beschreiben Sie (als Kommentar) die Funktionsweise von `init`; d.h. die einzelnen Schritte sowie die Verwendung von `strtok`. Erläutern Sie anhand von `"Michael Meier,34,INF"`, wie sich der Eingabestring `eingabe` durch Aufruf der Funktion `init` verändert.

```
void init(struct Studi *s, char *eingabe)
{
    char *token;
    token = strtok(eingabe, ",");
    strcpy(s->name, token);

    token = strtok(NULL, ",");
    sscanf(token, "%d", &(s->punkte));

    token = strtok(NULL, ",");
```

```
    if (0 == strncmp(token, "EIT", 3))
    {
        s->studiengang = EIT;
    }
    else if (0 == strncmp(token, "INF", 3))
    {
        s->studiengang = INF;
    }
    else
    {
        s->studiengang = MATH;
    }
}
```

d) Schreiben Sie eine Funktion `void busyStudi(Studi* studis, int n, Fach f)` die aus einem Array von `n` Studierenden-Einträgen den Namen des Studierenden mit der höchsten Punktzahl aus einer Fachrichtung `f` ermittelt und ausgibt.

```
void busyStudi(struct Studi *studis, int n, enum Fach f)
{
    struct Studi *s = NULL;
    int max_points = INT_MIN;
    for (int i = 0; i < n; i++)
    {
        if ((studis->studiengang == f) && (max_points < studis->punkte))
        {
            s = studis;
            max_points = studis->punkte;
        }
        studis++;
    }
    if (s != NULL)
    {
        printf("Fleissigster Studi der EIT: %s\n", s->name);
    }
}
```

e) Schreiben Sie eine `main`-Funktion, die wie in Aufgabe 1 oben beschrieben, die zunächst die Daten der Studierenden zeilenweise einliest und ein Array von Strukturen des `Studi` füllt. Die Eingabe soll dabei max. 100 Einträge enthalten. Die Ausgabe des Programms soll den Namen des fleißigsten Studis des Fachbereichs EIT beinhalten.

*Beispiel:*

| *Eingabe:* | *Ausgabe:* |
|---|---|
| `Michael Maier,12,INF`<br>`Marja Martin,56,EIT`<br>`Berta Bubble,23,INF`<br>`Karl Klein,34,EIT`<br>`Ulli Schmitt,87,MATH`<br>`Yan Ying,34,EIT` | `Fleissigster Studi der EIT: Marja Martin` |

```
int main(void)
{
    struct Studi studis[N_STUDIS];
    int n = 0;
    char buffer[N] = {0};

    while (n < 100)
    {
        char *s = fgets(buffer, N, stdin);
        if (s == NULL)
        {
            break;
        }
        init(&studis[n], s);
        n++;
```

```
    }
    busyStudi(studis, n, EIT);
    return 0;
}
```

## Aufgabe 3: Die Elfen und die Weihnachtsfeier (5 Punkte)

Die Elfen Anton, Berta, Claus und Doris freuen sich schon alle riesig auf das große Weihnachtsfest im Weihnachtsdorf. Beim Aufstellen des Tannenbaums kommt es aber zum großen Streit unter den Elfen: Steht der Baum gerade oder nicht? Der Baum wird kürzer und kürzer gesägt, bis schließlich nur noch wenige Nadeln übrig bleiben. Dem Weihnachtsmann platzt der Kragen: "Zur Strafe dürft ihr dieses Jahr nicht mitfeiern!" In einer gemeinsamen Diskussion kann man sich schließlich auf die folgenden Grundsätze verständigen:

- Mindestens ein Elf geht zu der Feier.
- Anton geht auf keinen Fall zusammen mit Doris.
- Wenn Berta geht, dann geht Claus mit.
- Wenn Anton und Claus gehen, dann bleibt Berta zu Hause.
- Wenn Anton zu Hause bleibt, dann geht entweder(!) Doris oder Claus (aber nicht beide).

Helfen Sie den Elfen, in dem Sie ein Programm schreiben, dass alle erlaubten Konstellationen ermittelt, in denen die Elfen am Fest teilnehmen können. Die Ausgabe des Programs soll pro Zeile eine erlaubte Konstellation sein, wobei die Namen der teilnehmenden Elfen immer aufsteigend sortiert ausgegeben werden sollen. Auf der Vorlesungshomepage finden Sie eine entsprechende Vorlage `party.c`.

```
#include <stdio.h>

enum bool
{
    FALSE,
    TRUE
};

int main(void)
{
    enum bool a, b, c, d;
    enum bool ok;

    for (a = FALSE; a <= TRUE; a++)
        for (b = FALSE; b <= TRUE; b++)
            for (c = FALSE; c <= TRUE; c++)
                for (d = FALSE; d <= TRUE; d++)
                {
                    ok = (a || b || c || d) &&
                         !(a && d) &&
                         (!b || c) &&
                         (!(a && c) || !b) &&
                         (a || ((c && !d) || (!c && d)));
                    if (ok)
                        printf("%s%s%s%s\n", a ? "Anton " : "",
                               b ? "Berta " : "",
                               c ? "Claus " : "",
                               d ? "Doris " : "");
                }
}
```

## Aufgabe 4: Ein toller Start ins Neue Jahr! (10 Punkte)

Das Mondprogramm der Volksrepublik China plant am Freitag, den 07.12.2018, die Sonde "Chang'e 4" zu starten, die auf der Rückseite des Mondes landen soll um die bislang wenig erforschte Gegend zu untersuchen. Sie wurden ausgewählt für diese wichtige Mission ein Programm zu entwickeln, das es erlaubt verschiedene Szenarien bei der Landung zu simulieren.

Nehmen Sie an, die Landefähre befindet sich in 50 km Höhe über der Mondoberfläche und hat eine Abwärtsgeschwindigkeit von derzeit $3.600km/Std = 1.000m/s$. Wenn Sie keinen Schub geben, so werden Sie mit $1,63m/s^2$ vom Mond angezogen. Die Antriebe Ihrer Landefähre vermögen bei vollem Schub jedoch eine Gegenbeschleunigung von $12m/s^2$ zu erzeugen und verbrauchen dabei pro Sekunde 100 Liter Treibstoff. Anfangs verfügen Sie über 10000 Liter Treibstoff. Ihre Landefähre kann eine Aufsetzgeschwindigkeit von $10m/s$ aushalten, ansonsten ist das Landemanöver gescheitert.

Es gelten folgende Formeln:

$$\begin{aligned} a &= a_{Mond} + a_{Schub} \\ v &= v_0 + a * t \\ s &= v_0 * t + 0.5at^2 \end{aligned}$$

Dabei bezeichnet $v$ die Geschwindigkeit, $a$ die Beschleunigung und $t$ die Zeit. Die Beschleunigungen durch die Mondanziehung $a_{Mond}$ und die Triebwerke $a_{Schub}$ addieren sich, wobei auf die Richtung zu achten ist. Der erste Term in der Gleichung für $s$ berechnet die zurückgelegte Wegstrecke bei konstanter Geschwindigkeit, der zweite Term die zusätzlich zurückgelegte Wegstrecke aufgrund der Beschleunigung.

Entwickeln Sie eine einfache Simulation für die Mondlandung und gehen Sie dazu wie folgt vor:

a) Definieren Sie die in der Aufgabenstellung angegebenen Werte als symbolische Präprozessorkonstanten.

b) Definieren Sie einen Struktur-Datentyp `sim_daten`, der die notwendigen Simulationsdaten (Simulationszeit, Höhe, Geschwindigkeit, Treibstoff) zusammenfasst.

c) Implementieren Sie eine Funktion `void init(struct sim_daten *data)`, die die Komponenten von `data` mit den Präprozessorkonstanten initialisiert.

d) Implementieren Sie eine Funktion `void ausgabe(struct sim_daten *data)`, die die Komponenten von `data` strukturiert ausgibt.

e) Implementieren Sie eine Funktion, die einen Simulationsschritt berechnet und die Komponenten von `data` aktualisiert:

```
void simulations_schritt(struct sim_daten *data, bool bremsschschub)
```

Der Parameter `bremsschub` gibt an, ob im aktuellen Simulationsschritt die Bremstriebwerke zünden. Ein Simulationsschritt hat eine Dauer von 1s.

f) Implementieren Sie schließlich eine `main`-Funktion. Diese besteht aus einer Schleife, die von der Kommandozeile einliest, ob im nächsten Simulationsschritt gebremst werden soll (Eingabe: 'y'/'n') und dann entsprechend die Funktionen `simulations_schritt` aufruft. Die Schleife wird abgebrochen, wenn entweder 'q' eingegeben wird oder die Sonde auf dem Mond aufgekommen ist.

g) Zum Abschluss ist auszugeben, ob der Landeversuch erfolgreich war oder scheiterte.

*Beispiel:*
*Eingabe:*

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

*Ausgabe:*

```
Die Sonde ist auf dem Mond zerschellt!
Zeit:                  49 s
Hoehe:               -956.8 m
Geschwindigkeit:     1079.9 m/s
Treibstoff:         10000.0 l
```

*Hinweis:*

- Die Ausgabe der Simulationsdaten soll mit 8 Vor- und 1 Nachkommastelle erfolgen. Die Angabe der Zeit ist als ganze Zahl gegeben.
- Bei erfolgreicher Landung ist der Ausgabetext: `Die Sonde ist erfolgreich auf dem Mond gelandet!`

- Bei missglückter Landung ist der Ausgabetext: `Die Sonde ist auf dem Mond zerschellt!`
- Bei Abbruch der Simulation soll ausgegeben werden: `Die Simulation wurde abgebrochen.`

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define BESCHL_MOND 1.63
#define BESCHL_TRIEBWERK 12.00

#define ANF_HOEHE 50000
#define ANF_GESCHW 1000
#define ANF_TREIBSTOFF 10000
#define TREIBSTOFFVERBRAUCH 100
#define AUFSETZ_GESCHW 10

struct sim_daten
{
    int zeit;
    double hoehe;
    double geschwindigkeit;
    double treibstoff;
};

void init(struct sim_daten *data);
void ausgabe(struct sim_daten *data);
void simulations_schritt(struct sim_daten *data, bool bremsschschub);

int main(void)
{
    struct sim_daten d;
    char eingabe;

    init(&d);

    do
    {
        eingabe = getchar();
        switch (eingabe)
        {
        case 'q':
            printf("Die Simulation wurde abgebrochen.\n");
            return 0;
        case 'y':
            simulations_schritt(&d, true);
            break;
        case 'n':
            simulations_schritt(&d, false);
            break;
        }
    } while (d.hoehe > 0);

    if (d.geschwindigkeit > AUFSETZ_GESCHW)
    {
        printf("Die Sonde ist auf dem Mond zerschellt!\n");
    }
    else
    {
        printf("Die Sonde ist erfolgreich auf dem Mond gelandet!\n");
    }
    ausgabe(&d);
    return 0;
}

void init(struct sim_daten *data)
{
    data->zeit = 0;
    data->hoehe = ANF_HOEHE;
    data->geschwindigkeit = ANF_GESCHW;
    data->treibstoff = ANF_TREIBSTOFF;
}

void ausgabe(struct sim_daten *data)
{
```

```
    printf("Zeit:             %8d s\n", data->zeit);
    printf("Hoehe:            %8.1f m\n", data->hoehe);
    printf("Geschwindigkeit: %8.1f m/s\n", data->geschwindigkeit);
    printf("Treibstoff:       %8.1f l\n", data->treibstoff);
}

void simulations_schritt(struct sim_daten *data, bool bremsschub)
{
    float beschleunigung;

    if (bremsschub && data->treibstoff > 0)
    {
        beschleunigung = (BESCHL_MOND - BESCHL_TRIEBWERK);
        data->treibstoff = data->treibstoff - TREIBSTOFFVERBRAUCH;
    }
    else
    {
        beschleunigung = BESCHL_MOND;
    }
    data->hoehe = data->hoehe - data->geschwindigkeit * 1 // v = a * t
                  - 0.5 * beschleunigung * 1 * 1;         // s = 1/2 a * t*t
    data->geschwindigkeit = data->geschwindigkeit + beschleunigung * 1;
    data->zeit++;
}
```