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

TU Kaiserslautern
Fachbereich Informatik

Lösungshinweise zum Übungsblatt 8: Programmieren in C (WS 2018/19)

1. Zur Beantwortung von Fragen und Hilfe bei Problemen stehen wir Ihnen immer in der Praktische Übung, Mittwoch 15:30, Raum 32-410-PC zur Verfügung sowie per Email an `pinc-support@cs.uni-kl.de`!
2. Sie können sich im Exclaim System unter `https://softech.cs.uni-kl.de/exclaim` mit Ihrem persönlichen STATS-Account einloggen und Dateien zu den einzelnen Übungen hochladen.
3. Die Aufgaben auf diesem Übungsblatt sind alle freiwillig zu bearbeiten.

Aufgabe 1: Schleifen

Abgabe: `schleifen.c` in Exclaim

Schreiben Sie ein Programm, das als Eingabe eine ungerade Zahl N einliest und folgende Ausgaben erstellt (hier für N=9):

```
000010000
000111000
001111100
011111110
111111111
```

N ist dabei die Anzahl der Zeichen pro Zeile. Schreiben Sie Ihre Lösung in zwei Varianten: mit einer `for`- und mit einer `while`- Schleife. [Warum ist eine Lösung mit einer `do-while` Schleife nicht gut umsetzbar?]. Die Ausgabe soll dann das Muster zweimal enthalten, mit einer Leerzeile getrennt.

Aufgabe 2: Arbeitsweise des Präprozessors verstehen

Abgabe: -

Gegeben sei folgender Programmcode vor dem Präprozessorlauf:

```
1  #include <stdio.h>
2  #define EINS 1
3  #define ZWEI 2
4  #define DREI 3
5  #define ZWEI_MAL_DREI ZWEI *DREI
6  #define MAL_ZWEI(x) x * 2
7  #define START main
8
9  int START(void)
10 {
11      int i, j, k, l, m, x;
12      i = EINS;
13      j = ZWEI * 5;
14      k = EINS + ZWEI;
15      l = ZWEI_MAL_DREI;
16 #undef ZWEI
17 #define ZWEI 6
18      m = j - ZWEI_MAL_DREI;
19      printf("Die Zahl EINS \n");
```

```
20      if (2 * MAL_ZWEI(3 + 1) != 16)
21      {
22          printf("???")
23      }
24
25      return (0);
26  }
```

Notieren Sie, wie der Programmcode nach dem Präprozessorlauf aussehen wird. Wie muss man die Präprozessor-Direktiven ändern, damit die Ausgabe in Zeile 22 verhindert wird?

```
int main(void)
{
    int i, j, k, l, m, x;
    i = 1;
    j = 2 * 5;
    k = 1 + 2;
    l = 2 * 3;

    m = j - 6 * 3;
    printf("Die Zahl EINS \n");
    if (2 * 3 + 1 * 2 != 16)
    {
        printf("???")
    }

    return (0);
}
```

Wenn man Präprozessor-Ausdrücke als abkürzende Notation verwendet, sollten sowohl alle Parameter einzeln, als auch der Ausdruck selbst geklammert sein.

```
#define MAL_ZWEI(x) ((x) * 2)
```

Aufgabe 3: Noch mehr Arrays

Abgabe: `gemeinsam.c` in Exclaim

Schreiben Sie ein C-Programm, das zwei Folgen von N int-Werten einliest und die Anzahl der Zahlen ausgibt, die in beiden Folgen vorkommen. Sie können davon ausgehen, dass keine Zahl in einer der Folgen mehrmals vorkommt.

Hinweis: Definieren Sie N als Präprozessor-Konstante N mit Wert 10 für die Testfälle in Exclaim. Führen Sie Ihre Implementierung von `gemeinsam.c` schrittweise im Debugger aus, wie in der Vorlesung vorgeführt. Setzen Sie dazu geeignet Breakpoints an die jeweiligen Schleifen, um die Veränderung der Variablenwerte in den einzelnen Schleifendurchlaufen nachzuvollziehen.

Beispiel: Bei Eingabe von

```
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 3 2 1
```

soll folgende Ausgabe `3` erfolgen.

```
#include <stdio.h>
#define N 10

int main(void)
{
    int a[N];
    int b[N];

    // Einlesen der Arrays
    for(int i = 0; i < N; i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i = 0; i < N; i++)
```

```
    {
        scanf("%d",&b[i]);
    }

    int count = 0;
    for(int i = 0; i < N; i++)
    {
        for(int j = 0; j < N; j++)
        {
            if(a[i] == b[j])
            {
                count++;
                break;
            }
        }
    }

    printf("%d\n", count);
    return 0;
}
```

Aufgabe 4: Noch mehr Strings

Abgabe: `str.c` in Exclaim

Schreiben Sie ein C-Programm, das einen String von der Konsole einliest und folgende Information ermittelt:

- Länge des Strings
- Ist der String ein Palindrom?

Verwenden Sie dazu **keine** Bibliotheksfunktionen aus der String-Bibliothek. Definieren die maximale Länge des Strings als Präprozessor-Konstante mit Wert 61, um genügend Speicherplatz für max. 60 Zeichen plus dem Stringterminalsymbol `\0` bereitzuhalten.

Beispiel: Bei Eingabe von `C ist die beste Programmiersprache der Welt` soll folgende Ausgabe erfolgen:

```
Der String hat die Laenge 43 und ist kein Palindrom.
```

Bei Eingabe von `RELIEFPFEILER` soll folgende Ausgabe erfolgen:

```
Der String hat die Laenge 13 und ist ein Palindrom.
```

Hinweis: Bei der Eingabe von Strings auf der Kommandozeile wird bei Verwendung von `fgets` der Zeilenumbruch als letztes Zeichen dem Eingabestring hinzugefügt. Um Ihre Lösung lokal zu testen, sollten Sie Ihren Eingabestring in einer Textdatei (z.B. `eingabe.txt` abspeichern und diese bei der Ausführung auf Kommandozeile folgendermaßen als Eingabe nutzen:

```
gcc str.c
./a.out < eingabe.txt
```

```
#include <stdio.h>
#include <stdbool.h>
#define N 61

int main(void)
{
    char eingabe[N] = {0};
    fgets(eingabe, N, stdin);

    // Ermitteln der Laenge
    int laenge = 0;
    char *pc = eingabe;
    while(*pc != '\0')
    {
        laenge++;
        pc++;
    }
```

```
    // Ist die Eingabe ein Palindrom?

    int i = 0;
    int j = laenge-1;
    bool isPalindrom = true;

    while (i<j)
    {
        if (eingabe[i] != eingabe[j])
        {
            isPalindrom = false;
            break;
        }
        i++;
        j--;
    }

    char* p;
    if (isPalindrom)
    {
        p = "ein";
    } else {
        p = "kein";
    }

    // Ausgabe
    printf("Der String hat die Laenge %d und ist %s Palindrom.\n", laenge, p);
    return 0;
}
```