

# Programmieren 1 – WS 2018/19

Prof. Dr. Michael Rohs, Tim Dünke, M.Sc.

## Übungsblatt 7

Alle Übungen (bis auf die erste) müssen in Zweiergruppen bearbeitet werden. Beide Gruppenmitglieder müssen die Lösung der Zweiergruppe einzeln abgeben. Die Namen beider Gruppenmitglieder müssen sowohl in der PDF Abgabe als auch als Kommentar in jeglichen Quelltextabgaben genannt werden. Wenn Sie für Übungsblatt 1 noch keinen Gruppenpartner haben, geben Sie alleine ab und nutzen Sie das erste Tutorium dazu, mit Hilfe des Tutors einen Partner zu finden. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 06.12. um 23:59 Uhr über <https://assignments.hci.uni-hannover.de/WiSe2018/Programmieren1>. Die Abgabe muss aus einer einzelnen zip-Datei bestehen, die den Quellcode, ein pdf bei Freitextaufgaben und alle weiteren nötigen Dateien (z.B. Eingabedaten oder Makefiles) enthält. Lösen Sie Umlaute in Dateinamen bitte auf.

Die Dokumentation der Prog1lib finden Sie unter: <https://hci.uni-hannover.de/files/prog1lib/index.html>

### Aufgabe 1: Verschachtelte Schleifen

Implementieren Sie Funktionen, die Ausgaben der angegebenen Form erzeugen. Der Parameter  $n$  beschreibt die Größe der Ausgabe. Erlaubte Werte für den Parameter  $n$  liegen im Intervall  $[0, 9]$ . In den Beispielen gilt  $n = 5$ . Verwenden Sie in dieser Aufgabe verschachtelte for-Schleifen. Die Template-Datei für diese Aufgabe ist loops.c.

a) Implementieren Sie die Funktion `void loops_a(int n)`, die Ausgaben der folgenden Form erzeugt:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

b) Implementieren Sie die Funktion `void loops_b(int n)`, die Ausgaben der folgenden Form erzeugt:

```
1
 1 2
 1 2 3
 1 2 3 4
1 2 3 4 5
```

c) Implementieren Sie die Funktion `void loops_c(int n)`, die Ausgaben der folgenden Form erzeugt:

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

d) Implementieren Sie die Funktion `void loops_d(int n)`, die Ausgaben der folgenden Form erzeugt:

```

X      X
  X    X
    X
  X    X
X      X

```

e) Implementieren Sie die Funktion `void loops_e(int n)`, die Ausgaben der folgenden Form erzeugt (für  $n \geq 3$ ). Sie dürfen beliebige Hilfsfunktionen implementieren.

```

+-----+
/       /
/       /
/       /
+-----+

```

f) Implementieren Sie die Funktion `void loops_f(int n)`, die Ausgaben der folgenden Form erzeugt (für  $n \geq 3$ ). Sie dürfen beliebige Hilfsfunktionen implementieren.

Beispiel für  $n = 5$ :

```

+-----+
/A B C D /
/E F G H /
/I J K L /
+-----+

```

Beispiel für  $n = 9$ :

```

+-----+
/A B C D E F G H /
/I J K L M N O P /
/Q R S T U V W X /
/Y Z A B C D E F /
/G H I J K L M N /
/O P Q R S T U V /
/W X Y Z A B C D /
+-----+

```

## Aufgabe 2: Währungen umrechnen Bitcoin, Euro, Dollar & Dänische Krone

Die Template-Datei für diese Aufgabe ist `money.c`. Die Struktur `Money` repräsentiert eine Menge von Geld in einer bestimmten Währung. Die möglichen Währungen (Bitcoin, Dollar, Euro und Dänische Krone) sind über die Aufzählung `Currency` definiert. Die Umrechnungskurse sind in der Template-Datei angegeben. Gegeben sind außerdem eine Konstruktorfunktion `make_money` und eine Testfunktion `test_within_value` für `Money` Strukturen.

a) Implementieren Sie die Funktion `print_money`, die Geld in Abhängigkeit der Währung in folgendem Format ausgibt:

```
1234.12 $
4.75423414 Bitcoin // <- 8 Nachkommastellen
33.30 Euro
70.75 DKK
```

b) Implementieren Sie die Funktion `Money to_Currency(Money m, Currency target_currency)`. Diese soll Geld in einer bestimmten Währung in eine Zielwährung umrechnen. Fügen Sie zunächst mindestens 5 sinnvolle Tests zu `to_currency_test` hinzu.

c) Implementieren Sie die Funktion `int compare(Money m, Money v)`. Diese soll zwei Mengen an Geld vergleichen und `0` zurückgeben, wenn `m` und `v` den gleichen Wert repräsentieren (Toleranz 0.01), `-1` zurückgeben, wenn `m` einen kleineren Wert als `v` hat und `1` zurückgeben, wenn `w` einen größeren Wert als `v` hat. Fügen Sie zunächst mindestens 5 Tests zu `compare_test` hinzu.

## Aufgabe 3: Dateien einlesen und verarbeiten

In dieser Aufgabe soll die Tabelle `people.txt` eingelesen und verarbeitet werden. Die Tabelle enthält eine Zeile mit Spaltenbeschreibungen und dann eine größere Anzahl Zeilen mit entsprechenden Werten. Die erste Spalte enthält Geburtsjahre (Typ `int`), die zweite die Geschlechter ('m', 'f') (Typ `char`), und die letzte Körpergrößen in Meter (Typ `double`). Ihr Programm soll alle enthaltenen Daten einlesen können. Berechnen Sie dann diese statistischen Angaben:

- Das durchschnittliche Geburtsjahr (gerundet auf ganze Jahre)
  - Die Anzahl Männer in der Tabelle
  - Die Anzahl Frauen in der Tabelle
  - Die durchschnittliche Körpergröße der Männer
  - Die durchschnittliche Körpergröße der Frauen
- a) Entwerfen Sie eine Struktur namens `Statistics`, die die genannten Angaben repräsentieren kann.
- b) Schreiben Sie eine Konstruktorfunktion `make_statistics`, welche die Komponenten der Struktur auf `0` bzw. `0.0` initialisiert.

- c) Schreiben Sie eine Funktion `print_statistics`, welche die Komponenten der Struktur in folgender Form auf der Konsole ausgibt:
- ```
mean year:          1921
number males:       12
number females:     28
mean height males:  1.28 m    (zwei Nachkommastellen)
mean height females: 2.17 m    (zwei Nachkommastellen)
```
- d) Schreiben Sie eine Funktion `compute_statistics`, welche die in einer Zeichenkette (Typ: `String`) vorliegende Tabelle verarbeitet und dafür eine Statistik berechnet und zurückgibt. Schauen Sie sich zunächst an, wie der Inhalt von `people.txt` formatiert ist. Ihre Funktion muss nur für dieses Format funktionieren. Eine Fehlerbehandlung soll nicht implementiert werden. Benutzen Sie zur Verarbeitung des Strings die Funktionen `s_length`, `s_get` und `s_sub`. Benutzen Sie zur Konvertierung eines Strings in ein `int` die Funktion `i_of_s` und zur Konvertierung eines Strings in ein `double` die Funktion `d_of_s`.

#### Aufgabe 4: Streichelzoo

In einem Programm sollen verschiedene Arten von Tieren mit bestimmten Attributen repräsentiert werden – nämlich Hunde, Katzen und Weiße Haie. Alle Tiere haben einen Namen und ein Gewicht.

- Katzen brauchen zusätzlich noch die Angabe, ob sie Wolle mögen.
- Hunde haben als zusätzliches Attribut ihre Halsband-Länge in Zentimeter.
- Weiße Haie haben als zusätzliche Attribute die Anzahl an gefressenen Surfern und die Angabe, ob sie gerade hungrig sind.

Entwickeln Sie eine Funktion, die ein Tier als Eingabe bekommt und einen entsprechenden String auf der Konsole ausgibt, welcher das Tier beschreibt.

Beispielsweise könnte eine Ausgabe folgendermaßen aussehen:

```
"Hello there, I'm a white shark, my name is Henry and I weigh 1530
kg. I'm quite proud of having eaten 42 surfers on my journeys. Oh
and I'm hungry right now. Yummy, I can see a human! Better start
swimming real fast ..."
```

Verwenden Sie die im Skript unter *Recipe for Variant Data* beschriebenen Schritte. Verwenden Sie die Template-Datei `animals.c`. Passen Sie den Inhalt der Datei an die Aufgabe an, aber behalten Sie das generelle Layout bei. Generieren Sie für jede Tierart je drei sinnvolle Beispiele und geben Sie diese mit Hilfe Ihrer Funktion auf der Konsole aus.

Hinweis:

- Die Funktion `printf(templateString, arg1, arg2, ...)` akzeptiert im `templateString` zahlreiche Platzhalter für verschiedene Argumente. `%s` für `String`, `%f` für `double`, `%i` für `integer` usw. Mit Hilfe von `%.2f` können Sie beispielsweise einen `double` Wert mit zwei Nachkommastellen ausgeben. Weitere Informationen finden Sie in der Dokumentation von `printf`: <http://en.cppreference.com/w/cpp/io/c/fprintf>

Hinweise zum Editieren, Compilieren und Ausführen:

- mit Texteditor `file.c` editieren und speichern
- `make file` ← ausführbares Programm erstellen
- `./file` ← Programm starten (evtl. ohne `./`)
- Die letzten beiden Schritte lassen sich auf der Kommandozeile kombinieren zu:  
`make file && ./file`