



Universität Augsburg
Institut für Informatik

Vorkurs Informatik

Fakultät für Angewandte Informatik

Lehrprofessur für Informatik

PROF. DR. LORENZ

Übungblatt Thema 5

Allgemeine Hinweise:

- Achten Sie bei allen Programmieraufgaben auf *Kompilierbarkeit* und *Einhaltung der Coding Conventions* (werden zu jedem Thema in den Vorlesungsfolien erklärt); auch dann, wenn es nicht explizit im Aufgabentext gefordert ist.

- Gehen Sie davon aus, dass alle Programme in der Programmiersprache C zu erstellen sind.

- Kompilieren Sie alle Ihre Programme mit den folgenden *Compiler-Schaltern*:

`-Wall -Wextra -std:c11 -pedantic`

Achten Sie darauf, dass trotz Verwendung dieser Schalter keine Fehler-/Warnmeldungen erzeugt werden.

In den folgenden Programmieraufgaben werden Sie sich mit

- dem Schreiben eigener Funktionen,

Zur Eigenrecherche können u.a. die folgenden Internetseiten dienen:

- https://de.wikibooks.org/wiki/C-Programmierung:_Funktionen

Aufgabe 1 (*Einfache Funktionen, ca. 30 Minuten*)

Erstellen Sie für jede Teilaufgabe jeweils eine C-Datei mit einer `main`-Funktion, einer weiteren Funktion gemäß der Aufgabenstellung und ggf. nützlichen Funktionen aus vorherigen Teilaufgaben. In der `main`-Funktion testen Sie jeweils die neue Funktion.

a) Erstellen Sie eine Funktion mit dem Prototyp

```
int intrand(int bound)
```

nach folgenden Vorgaben:

*Es soll eine zufällige ganze Zahl zwischen 0 und **bound** (jeweils einschließlich) berechnet und zurückgegeben werden.*

In der `main`-Funktion testen Sie die `intrand`-Funktion, indem Sie mit `intrand` mittels einer Schleife 20 zufällige ganze Zahlen zwischen 0 und 100 erzeugen und diese zeilenweise ausgeben.

b) Erstellen Sie eine Funktion mit dem Prototyp

```
double doublerand(void)
```

nach folgenden Vorgaben:

Es soll eine zufällige Dezimalzahl zwischen 0.0 und 1.0 (jeweils einschließlich) berechnet und zurückgegeben werden.

In der `main`-Funktion testen Sie die `doublerand`-Funktion, indem Sie mit `doublerand` mittels einer Schleife 20 zufällige Dezimalzahlen erzeugen und diese zeilenweise ausgeben.

c) Erstellen Sie eine Funktion mit dem Prototyp

```
double increment(double x)
```

nach folgenden Vorgaben:

*Die übergebene Dezimalzahl **x** soll um 1 erhöht werden und das Ergebnis zurückgegeben werden.*

In der `main`-Funktion testen Sie die `increment`-Funktion, indem Sie eine zufällige Dezimalzahl `x` aus dem Bereich von 0.0 bis 1.0 erzeugen, und `x` sowie `increment(x)` ausgeben.

d) Erstellen Sie eine Funktion mit dem Prototyp

```
void input_message(void)
```

nach folgenden Vorgaben:

Es soll folgende Nachricht auf Kommandozeile ausgegeben werden:

"Bitte geben Sie Ihr Geburtsjahr ein: "

In der `main`-Funktion testen Sie die `input_message`-Funktion, indem Sie einmal die `input_message`-Funktion aufrufen.

e) Erstellen Sie eine Funktion mit dem Prototyp

```
bool my_islower(int c)
```

nach folgenden Vorgaben (Nachimplementierung der `islower`-Funktion aus `ctype.h`):

-
- Ist der übergebene Wert `c` der ASCII-Code eines lateinischen Kleinbuchstaben, soll `true` zurückgegeben werden.
 - Ist der übergebene Wert `c` nicht der ASCII-Code eines lateinischen Kleinbuchstaben, soll `false` zurückgegeben werden.

In der `main`-Funktion testen Sie die `my_islower`-Funktion, indem Sie in einer Schleife 10 zufällige ganze Zahlen `c` zwischen 0 und 127 (jeweils einschließlich) erzeugen, diese jeweils als ASCII-Zeichen und das Ergebnis des Aufrufs `my_islower(c)` ausgeben.

Aufgabe 2 (Einfache Funktionen - Fortsetzung, ca. 30 Minuten)

Implementieren Sie nach dem Schema aus Aufgabe 1 folgende weitere Funktionen und testen Sie diese in einer `main`-Funktion. Einen geeigneten Prototyp der Funktion entwerfen Sie dabei jeweils selbst.

a) (Dekrement-Funktion)

Eine Funktion, die einen übergebenen Wert um 1 verringert und zurückgibt.

b) (Zufallszahlen - Fortsetzung)

Eine Funktion, die eine Zufallszahl aus einem vorgegebenen Intervall erzeugt und zurückgibt.

c) (is-Funktionen)

Weitere is-Funktionen aus `ctype.h`.

d) (to-Funktionen)

Die to-Funktionen aus `ctype.h`.

Aufgabe 3 (Fortgeschrittene Funktionen, ca. 60 Minuten)

Erstellen Sie für jede Teilaufgabe jeweils eine C-Datei mit einer `main`-Funktion, einer weiteren Funktion gemäß der Aufgabenstellung und ggf. nützlichen Funktionen aus vorherigen Teilaufgaben. Die Erstellung der neuen Funktion erfordert jeweils die Benutzung von geeigneten Schleifen. In der `main`-Funktion testen Sie jeweils die neue Funktion.

Hinweis: Für einige der Teilaufgaben können Sie Code aus den Aufgaben 3 und 4 von Übungsblatt 4 wiederverwenden.

a) Erstellen Sie eine Funktion mit dem Prototyp

```
void print_lines(int count)
```

nach folgenden Vorgaben:

- Falls `count <= 0` gilt, soll abgebrochen werden.
- Es sollen `count` Leerzeilen auf Kommandozeile ausgegeben werden.

In der `main`-Funktion testen Sie die `print_lines`-Funktion nach folgenden Vorgaben:

- Rufen Sie die `print_lines`-Funktion für den Übergabewert 10 auf.
- Geben Sie `x` in einer neuen Zeile aus.
- Rufen Sie die `print_lines`-Funktion für den Übergabewert 5 auf.

-
- Geben Sie `x` in einer neuen Zeile aus.
 - Rufen Sie die `print_lines`-Funktion für den Übergabewert `-1` auf.

b) Erstellen Sie eine Funktion mit dem Prototyp

```
void print_ascii(int lower, int upper)
```

nach folgenden Vorgaben:

- Falls `lower` und `upper` keine Werte zwischen 0 und 127 (jeweils einschließlich) sind, oder `lower > upper` gilt, soll abgebrochen werden.
- Es sollen zeilenweise die ASCII-Zeichen mit den ASCII-Codes zwischen `lower` und `upper` (jeweils einschließlich) auf Kommandozeile ausgegeben werden.

In der `main`-Funktion testen Sie die `print_ascii`-Funktion nach folgenden Vorgaben:

- Rufen Sie die `print_ascii`-Funktion auf, um die ASCII-Zeichen mit den ASCII-Codes zwischen 48 und 122 (jeweils einschließlich) auszugeben.
- Rufen Sie die `print_ascii`-Funktion mit Übergabewerten auf, von denen einer negativ ist.
- Rufen Sie die `print_ascii`-Funktion mit Übergabewerten auf, von denen einer größer als 128 ist.

c) Erstellen Sie eine Funktion mit dem Prototyp

```
void print_mult(int bound, int div)
```

nach folgenden Vorgaben:

- Falls `bound` und `div` keine positiven Werte sind, soll abgebrochen werden.
- Es sollen zeilenweise alle durch `div` teilbaren ganzen Zahlen zwischen 1 und `bound` (jeweils einschließlich) auf Kommandozeile ausgegeben werden.

In der `main`-Funktion testen Sie die `print_mult`-Funktion nach folgenden Vorgaben:

- Rufen Sie die `print_mult`-Funktion auf, um alle durch 17 teilbaren ganzen Zahlen zwischen 1 und 1000 (jeweils einschließlich) auszugeben.
- Rufen Sie die `print_mult`-Funktion mit Übergabewerten auf, von denen einer negativ ist.

d) Erstellen Sie eine Funktion mit dem Prototyp

```
void print_square(int width, int height, int c)
```

nach folgenden Vorgaben:

- Falls `width` und `height` keine positiven Werte sind, soll abgebrochen werden.
- Falls `c` kein Wert zwischen 0 und 127 (jeweils einschließlich) ist, soll abgebrochen werden.
- Es sollen `height` Zeilen aus je `width` Zeichen `c` auf Kommandozeile ausgegeben werden.

In der `main`-Funktion testen Sie die `print_square`-Funktion nach folgenden Vorgaben:

- Rufen Sie die `print_square`-Funktion auf, um 7 Zeilen aus je 12 Zeichen `'*'` auf Kommandozeile auszugeben.
- Rufen Sie die `print_square`-Funktion mit Übergabewerten auf, von denen einer negativ ist.

e) Erstellen Sie eine Funktion mit dem Prototyp

```
double factorial(int n)
```

nach folgenden Vorgaben:

- Es soll die Fakultät $n!$ (das ist das Produkt der ganzen Zahlen von 1 bis n) berechnet und als `double`-Wert zurückgegeben werden.

In der `main`-Funktion testen Sie die `factorial`-Funktion nach folgenden Vorgaben:

- Berechnen Sie in einer Schleife die Fakultäten der Zahlen von 1 bis 50 und geben Sie diese zeilenweise aus.

Aufgabe 4 (*Fortgeschrittene Funktionen - Fortsetzung, ca. 60 Minuten*)

Implementieren Sie nach dem Schema aus Aufgabe 3 folgende weitere Funktionen und testen Sie diese in einer `main`-Funktion. Einen geeigneten Prototyp der Funktion entwerfen Sie dabei jeweils selbst.

a) (Potenzen)

Erstellen Sie eine Funktion, die für eine nicht-negative ganze Zahl n die Potenz 2^n berechnet und zurückgibt (ohne Benutzung von Bibliotheksfunktionen).

b) (Primzahlen)

Erstellen Sie eine Funktion, mit der Sie testen können, ob eine positive ganze Zahl n eine Primzahl ist.

c) Die Aufgaben 3 und 4 von Übungsblatt 4 bieten weitere Möglichkeiten zur Erstellung von Funktionen.