



Universität Augsburg
Institut für Informatik

Vorkurs Informatik

Fakultät für Angewandte Informatik

Lehrprofessur für Informatik

PROF. DR. LORENZ

Übungsblatt Thema 4

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 Datentyp `double`,
- mathematische Funktionen der Bibliothek `math.h`,
- und Wiederholungen beschäftigen.

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

- http://de.wikibooks.org/wiki/C-Programmierung:_Einfache_Ein-_und_Ausgabe
- http://de.wikibooks.org/wiki/C-Programmierung:_Ausdr%C3%BCcke_und_Operatoren
- https://de.wikibooks.org/wiki/C-Programmierung:_Kontrollstrukturen#Schleifen
- https://de.wikibooks.org/wiki/C-Programmierung:_Standard_Header#math.h

In den Aufgaben 1 - 4 soll in jeder Teilaufgabe ein C-Programm mit einer eigenen `main`-Funktion erstellt werden.

Aufgabe 1 (Dezimalzahlen, ca. 15 Minuten)

a)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `-117.375` zu.
- Geben Sie den Wert von `x` in Festkommenschreibweise aus.
- Geben Sie den Wert von `x` in Fließkommenschreibweise aus.

b)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `0.0` zu.
- Geben Sie den Wert von `x` in Festkommenschreibweise aus.
- Geben Sie den Wert von `x` in Fließkommenschreibweise aus.

c)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `DBL_MIN` zu.
- Geben Sie den Wert von `x` in Festkommenschreibweise aus.
- Geben Sie den Wert von `x` in Fließkommenschreibweise aus.

d)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `100e2` zu.
- Geben Sie den Wert von `x` in Festkommenschreibweise aus.
- Geben Sie den Wert von `x` in Fließkommenschreibweise aus.

e)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `-100e-3` zu.
- Geben Sie den Wert von `x` in Festkommenschreibweise aus.
- Geben Sie den Wert von `x` in Fließkommenschreibweise aus.

f) Geben Sie folgende Werte jeweils in einer eigenen Zeile in Festkommenschreibweise aus, indem Sie jeweils geeignete Funktionen aus `math.h` benutzen:

- die Quadratwurzel aus `2.0`
- den Sinus von `1.0`
- den natürlichen Logarithmus von `1000.0`
- den Wert der Exponentialfunktion angewendet auf `5.0`
- den größten ganzzahligen Wert, der kleiner oder gleich der Quadratwurzel aus `1000.0` ist.
Benutzen Sie eine der Funktionen `ceil` oder `floor`.

Aufgabe 2 (Typumwandlung, Rundung und Overflow, ca. 30 Minuten)

Alle Ausgaben in dieser Aufgabe sollen in Festkommenschreibweise mit 10 Nachkommastellen erfolgen.

a)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `DBL_MAX` zu.
- Geben Sie den Wert von `x` aus.
- Erhöhen Sie dann den Wert von `x` um 1.
- Geben Sie wiederum den Wert von `x` aus.

Die Rechnung in Teilaufgabe a) führt zu einem sog. **Overflow**. Das bedeutet, dass der Wertebereich des benutzten Datentyps `double` verlassen wird.

b)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert `1e100` zu.
- Geben Sie den Wert von `x` aus.
- Erhöhen Sie dann den Wert von `x` um 1.
- Geben Sie wiederum den Wert von `x` aus.

Die Rechnung in Teilaufgabe b) kann zu einem **Rundungsfehler** führen, da hier mit Zahlen aus sehr unterschiedlichen Größenbereichen gerechnet wird. Die Hintergründe dazu erfährt man in der Vorlesung "Informatik 1".

c)

- Deklarieren Sie eine `double`-Variable `x`.
- Weisen Sie ihr den Wert 100 zu.
- Geben Sie den Wert von `x` aus.

d)

- Deklarieren Sie eine `int`-Variable `n`.
- Weisen Sie ihr den Wert 0.6 zu.
- Geben Sie den Wert von `n` aus.

In den Teilaufgaben c) und d) können Sie die automatische Typumwandlung von Werten bei Wertzuweisungen beobachten.

e)

- Deklarieren Sie eine `double`-Variable `x` und eine `int`-Variable `n`.
- Weisen Sie beiden Variablen den Wert 5.0 zu.
- Geben Sie den Wert beider Variablen aus.

-
- Dividieren Sie beide Variablen jeweils durch 2.
 - Geben Sie den Wert beider Variablen aus.

f)

- Geben Sie den Wert von 5 dividiert durch 2 aus.
- Geben Sie den Wert von 5 dividiert durch 2.0 aus.

In den Teilaufgaben e) und f) sehen Sie, dass das Ergebnis einer Division abhängig vom Typ der Werte ist.

g)

- Generieren Sie eine ganze Zufallszahl `n`.
- Berechnen Sie aus `n` eine zufällige Dezimalzahl `x` zwischen 0.0 und 1.0 mittels Typumwandlung nach `double` und Division durch `RAND_MAX`.
- Geben Sie `x` aus.

Auf diese Art und Weise lassen sich aus ganzen Zufallszahlen zufällige Dezimalzahlen berechnen.

Aufgabe 3 (Wiederholungen, ca. 60 Minuten)

a) Geben Sie mit einer `for`-Schleife 10 Leerzeilen aus, indem Sie pro Schleifendurchlauf eine Leerzeile ausgeben.

b) Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl, weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 101 zu, und geben Sie mit einer `while`-Schleife `n`-mal das Zeichen '`x`' aus.

Die Ausgabe sieht für `n==7` so aus:

```
xxxxxxx
```

c) Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl, weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 101 zu, und geben Sie mit einer `for`-Schleife zeilenweise die Zahlen von 1 bis `n` aus.

Die Ausgabe sieht für `n==5` so aus:

```
1
2
3
4
5
```

d) Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl, weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 1001 zu, und geben Sie mit einer `while`-Schleife alle Quadratzahlen zwischen 1 und `n` jeweils getrennt durch ein Komma aus.

Die Ausgabe sieht für `n==40` so aus:

```
1,4,9,16,25,36
```

e)

- Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl und weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 101 zu.
- Berechnen Sie mit einer `while`-Schleife die Summe aller ungeraden Zahlen zwischen 1 und `n`, indem Sie mit der Summe 0 beginnen und in jedem Schleifendurchlauf eine ungerade Zahl addieren.
- Geben Sie zeilenweise das Zwischenergebnis der Summe nach jedem Schleifendurchlauf aus.

Die Ausgabe sieht für `n==10` so aus:

```
1
4
9
16
25
```

f)

- Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl und weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 21 zu.
- Berechnen Sie mit einer `for`-Schleife das Produkt der Zahlen von 1 bis `n`, indem Sie mit dem Produkt 1 beginnen und in jedem Schleifendurchlauf eine Zahl hinzumultiplizieren.
- Geben Sie zeilenweise das Zwischenergebnis des Produkts nach jedem Schleifendurchlauf aus.

Anmerkung: Das berechnete Produkt nennt man **Fakultät von n** (in Zeichen: $n!$)

Die Ausgabe sieht für `n==6` so aus:

```
1
2
6
24
120
720
```

g)

- Berechnen Sie das Produkt von 21 und 37, indem Sie mit einer `for`-Schleife 37-mal die Zahl 21 addieren (also ohne Benutzung des `*` - Operators).
- Geben Sie zeilenweise das Zwischenergebnis der Addition nach jedem Schleifendurchlauf aus.

Die Ausgabe beginnt so:

```
21
42
63
84
```

h)

- Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl und weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 10001 zu.
- Berechnen Sie mit einer `while`-Schleife, wie oft `n` ganzzahlig durch 2 dividiert werden kann, bis das Ergebnis kleiner oder gleich 1 ist.

- Geben Sie zeilenweise das Zwischenergebnis der Division nach jedem Schleifendurchlauf und am Ende die Anzahl der Schleifendurchläufe aus.

*Anmerkung: Die berechnete Zahl nennt man **ganzzahligen Logarithmus (zur Basis 2) von n***

Die Ausgabe sieht für `n==100` so aus:

```
50
25
12
6
3
1
6
```

i)

- Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl und weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 10001 zu.
- Berechnen Sie mit einer `for`-Schleife, ob `n` eine Primzahl ist, indem Sie für alle ganzen Zahlen zwischen 2 und der Quadratwurzel von `n` testen, ob diese Teiler von `n` sind.
- Geben Sie eine entsprechende Textmeldung aus.

Anmerkung: Eine Zahl ist eine Primzahl, falls sie nur durch 1 und sich selbst (ohne Rest) teilbar ist

Aufgabe 4 (Verschachtelte Schleifen, ca. 45 Minuten)

a) (Rechteck ausgeben)

Geben Sie mit zwei ineinander verschachtelten `for`-Schleifen 10 Zeilen aus (äußere Schleife), die jeweils aus 20 hintereinander geschriebenen `'x'`-Zeichen bestehen (innere Schleife), indem Sie pro Schleifendurchlauf der inneren Schleife einmal das Zeichen `'x'` ausgeben.

Die Ausgabe sieht so aus:

```
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
```

b) (Dreieck ausgeben)

Geben Sie mit zwei ineinander verschachtelten `for`-Schleifen 10 Zeilen aus (äußere Schleife), wobei die `i`-te Zeile aus `i` hintereinander geschriebenen `'x'`-Zeichen bestehen (innere Schleife), indem Sie pro Schleifendurchlauf der inneren Schleife einmal das Zeichen `'x'` ausgeben.

Die Ausgabe sieht so aus:

```
x
xx
xxx
```

```
xxxx
xxxxx
xxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
```

c)

- Deklarieren Sie eine `int`-Variable `n`, generieren Sie eine ganze Zufallszahl und weisen Sie `n` als Wert den Rest bei ganzzahliger Division der Zufallszahl durch 31 zu.
- Geben Sie mit zwei ineinander verschachtelten `for`-Schleifen für jede Zahl `k` zwischen 1 und `n` (äußere Schleife) jeweils die Summe der Zahlen von `k` bis `2·k` (innere Schleife) aus.

Die Ausgabe beginnt so:

```
3
9
18
30
45
```

d) Geben Sie mit zwei ineinander verschachtelten `for`-Schleifen zeilenweise alle Primzahlen zwischen 2 und 1000 aus.

Die Ausgabe beginnt so:

```
2
3
5
7
11
```

e) Geben Sie mit zwei ineinander verschachtelten `for`-Schleifen eine übersichtliche 10er 1-mal-1 Tabelle aus (also eine Tabelle mit den Produkten von `1 * 1` bis `10 * 10`).

Die Tabelle beginnt so:

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30