Prof. Dr. Hauke Schramm
Fachbereich Informatik & Elektrotechnik
Fachhochschule Kiel

Programmieren in C

Aufgabenblatt 2

- **Abgabe in 2 Wochen (es gilt das in Moodle genannte Abgabedatum)**
- **Die Aufgaben 1 und 2 sind freiwillig – 3 und 4 müssen per Moodle abgegeben werden.**

Aufgabe 1 (Variablen und Konstanten) **Freiwillig!**

Markieren Sie alle „fehlerhaften“ Zeilen in folgendem Rumpfprogramm (der Begriff „Fehler“ umfasst hier Compilerfehler, ggf. Compilerwarnungen und vor allem inhaltliche Fehler, die nicht notwendig zu Compilerfehlern oder Compilerwarnungen führen). Begründen Sie, warum es sich jeweils um einen Fehler handelt und korrigieren Sie die entsprechenden Zeilen sinnvoll, so daß das Programm einwandfrei kompiliert und inhaltlich (im Sinne von Variablendefinitionen) sinnvoll ist.

```
#include <stdio.h>

int main(void)
{
    int i, j;
    float a = 2.3;
    int grösse = 5;
    char a = 'x';
    float f = 2.7e-5;
    const float Zinssatz;
    char b = 'ab';
    char c = ' \' ';
    const int n = 3;
    int grösse;

    {
          char x = 'Y';
          const char z = '\\';
          double umsatz-firma;
          int m;
    }

    m = 2.0;
    n = 5;
    Zinssatz = 1.5;
    f * f = f * 5;
    printf('Hallo Welt\n');
    return 0;
}
```

Aufgabe 2 (Einlesen von Zahlen, Summe berechnen, ausgeben) Freiwillig!

Wir haben in der Vorlesung das Einlesen von Zahlen und Buchstaben von der Tastatur und deren formatierte Ausgabe auf den Bildschirm kennengelernt. Dazu wurden die Funktionen scanf() und printf() verwendet. Setzen Sie diese Funktionen ein, um die folgende Aufgabe zu lösen:

Der Benutzer wird gebeten, 4 Gleitpunktzahlen, getrennt durch Leerzeichen, einzugeben. Das Programm berechnet die Summe der Zahlen und gibt die zugehörige mathematische Gleichung mit dem Ergebnis aus.

Beispiel für einen Programmlauf:

```
   > ./Aufgabe2.exe
Bitte geben Sie 4 Gleitpunktzahlen ein, getrennt durch (ein)
Leerzeichen:
2.1e-1 0.21 0.021e1 210e-3
0.210000 + 0.210000 + 0.210000 + 0.210000 = 0.840000
```

Testen Sie Ihr Programm, indem Sie die Zahl 0.21 in der Eingabezeile auf 4 verschiedene Arten (z.B. unter Verwendung der Exponentialschreibweise) angeben.

Die Lösung der Aufgabe soll den Quelltext und die Ausgabe des Programms umfassen. Die Programmausgabe können Sie als Kommentar in den Quellcode einfügen.

Hinweis: Mit dem Steuersymbol `%.2f` können Sie die Ausgabe einer float-Zahl mit nur 2 Nachkommastellen erzwingen.

Aufgabe 3 (Ein- und Auslesen verschiedener Formate):

Frau Schulze kauft auf dem Wochenmarkt ein. Sie benötigt Äpfel, Melonen und Pfirsiche. Schreiben Sie ein Programm, das den Händler bei der Preisberechnung unterstützt. Das Programm fragt die Kundin zunächst nach der Menge des jeweiligen Artikels und berechnet dann aus Menge und Einzelpreis den aktuellen Betrag (Ausgabe mit 2 Nachkommastellen!). Am Ende berechnet das Programm den Gesamtbetrag (Ausgabe ebenfalls mit 2 Nachkommastellen!). Benutzen Sie folgende Einzelpreise, die Sie als Konstante deklarieren:
1 kg Äpfel kostet 1.70 Euro
1 Melone kostet 1 Euro
1 kg Pfirsiche kostet 1.80 Euro

Der Programmlauf sollte folgendermaßen aussehen:

```
Wieviel kg Aepfel?
1.5
Das sind 2.55 Euro
Wieviele Melonen?
2
Das sind 2 Euro
Wieviele kg Pfirsiche?
2.5
Das sind 4.50 Euro
Das macht zusammen 9.05 Euro
```

Achten Sie hierbei insbesondere auf den richtigen Datentyp für die einzelnen Variablen!

Die Lösung soll das Programm und die Programmausgabe umfassen.

Aufgabe 4 (Ein- und Auslesen von Zahlen und Zeichen):

Schreiben Sie ein Programm, das den Benutzer auffordert, seinen Vornamen, Nachnamen, Geburtsdatum und Steuersatz einzugeben. Das Programm soll jedoch nur den ersten Buchstaben des Vornamens, den Nachnamen auf exakt 8 Buchstaben, das Geburtsdatum und den Steuersatz speichern, um diese Werte dann in einem gegebenen Format wieder auf dem Bildschirm auszugeben. Der Programmlauf sollte dabei wie folgt aussehen:

```
Bitte geben Sie Ihren Vornamen ein: Donald
Bitte geben Sie Ihren Nachnamen ein (mit Leerzeichen auf 8 Buchstaben): Duck

Bitte geben Sie Ihr Geburtsdatum im Format dd.mm.yyyy ein: 12.3.1876
Bitte geben Sie Ihren Steuersatz ein: 48.9

**********************
Name: D. Duck
Geburtsdatum: 12.3.1876
Steuersatz: 48.90
```

Hinweise:

- Lesen Sie die Eingabe des Vornamens mit **`scanf`** in eine einzelne **`char`**-Variable ein; danach sollte der (gesuchte) erste Buchstabe des Vornamens in dieser Variable stehen. Ein Problem ist, daß die weiteren Buchstaben des Vornamens nun noch im Puffer des Eingabestroms (Tastaturspeichers) stehen. Diese weiteren Buchstaben würden bei den folgenden **`scanf`**-Befehlen in die jeweiligen nächsten Variablen eingelesen und folglich Probleme bereiten. Daher können Sie (unter Windows) die zusätzlichen Buchstaben des Vornamens entfernen, indem Sie den Puffer des Eingabestroms nach der Eingabe des Vornamens mit
  `fflush(stdin);`
  leeren. Erläuterungen zu fflush finden Sie auch im Buch „C von A bis Z“ von Jürgen Wolf.
  ACHTUNG: Der Befehl **`"fflush()"`** funktioniert nicht auf Unix-Systemen! Dort können Sie mit **`clean_stdin()`** arbeiten.

- Der Nachname soll „vollständig“ eingelesen werden. Da wir noch keine Zeichenketten einlesen können, nehmen Sie bitte zur Vereinfachung an, daß der Nachname aus exakt 8 Buchstaben besteht. Falls der tatsächliche Nachname länger ist, werden die folgenden Buchstaben der Eingabe einfach ignoriert (also nach dem **`scanf`**-Befehl wieder – wie oben – **`fflush(stdin);`** einfügen). Falls der tatsächliche Nachname kürzer ist als 8 Buchstaben, müssen bei der Eingabe zusätzliche Leerzeichen eingefügt werden, so daß insgesamt genau 8 Zeichen eingegeben werden.

- Bei der Ausgabe soll der Name als <erster Buchstabe Vorname>. <Nachname> ausgegeben werden (Beispiel siehe oben). Der Nachname besteht bei der Ausgabe aus exakt den 8 gespeicherten Buchstaben des Nachnamens.

- Das Geburtsdatum soll im Format dd.mm.yyyy eingegeben und später auch wieder ausgegeben werden.