*Frankfurt University of Applied Sciences*
*Fachbereich 2: Informatik und Ingenieurwissenschaften*

# Einführung in die Programmierung mit der Programmiersprache C (**EifP**, WS2025/26)

Prof. Dr. Thomas Gabel
Roman Ahmad, Emre Özöner, Gina Romanazzi

# Aufgabenblatt 6

## Aufgabe 23: Mathematische Hilfsfunktionen

### Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Funktionen
- Funktionsaufrufe

### Aufgabenstellung

In dieser Aufgabe üben Sie das Erstellen und Nutzen von Funktionen.

a) Schreiben Sie ein C-Programm, das neben dem Hauptprogramm, also der `main()`-Funktion, noch über die folgenden Funktionen verfügt:

   - `float quadratEinerFliesskommazahl(float x)` - soll für den übergebenen Parameter $x$ dessen Quadrat $x^2$ zurückgeben
   - `int quadratEinerGanzzahl(int n)` - soll für den übergebenen ganzzahligen Parameter $n$ dessen Quadrat $n^2$ zurückgeben
   - `int summeDerErstenNNatuerlichenZahlen(int n)` - soll für den übergebenen positiven Parameter $n$ die Summe der natürlichen Zahlen von 0 bis $n$ zurückgeben, d.h. Rückgabewert ist $\sum_{i=0}^{n} i$ (vgl. Aufgabe 5 vom Aufgabenblatt 2[1]); sollte die Funktion mit einer negativen Zahl als Parameter aufgerufen werden ($n < 0$), so soll eine entsprechende Fehlermeldung ausgegeben werden und die Funktion 0 zurückliefern
   - `void summeDerNatuerlichenZahlenVonMBisN(int m, int n)` - soll für die beiden übergebenen positiven Parameter $m$ und $n$ (mit $m \leq n$) die Summe der natürlichen Zahlen von $m$ bis $n$ berechnen und ausgeben (*nicht* zurückgeben). Diese Funktion sollte dabei die im vorigen Punkt entwickelte Funktion aufrufen. Falls $m$

[1] Die aus der Mathematik bekannte Gauß'sche Summenformel darf hier verwendet werden.

oder $n$ negativ sind oder aber $m > n$, so soll eine Fehlermeldung ausgegeben werden.

b) Befüllen Sie das Hauptprogramm so, dass alle definierten Funktionen gut durch Sie getestet werden können. Hierbei sollten vom Benutzer Werte eingelesen, für diese die definierten Funktionen aufgerufen und die berechneten Ergebnisse ausgegeben werden.

c) Ergänzen Sie Ihr Programm um die Liste der Funktionsdeklarationen (Prototypen) aller von Ihnen selbst definierten und programmierten Funlktionen (vgl. Vorlesungsfolie "Funktionen (5)").

## Aufgabe 24: Einfache Schleifen

### Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Iteration mittels Schleifen

### Aufgabenstellung

Schreiben Sie ein Programm `hello24`, das den Namen des Benutzers erfragt und 24 mal auf den Bildschirm "Hello , <Benutzername>" ausgibt.
*Hinweis:* Verwenden Sie hierbei Ihre Lösung zu Aufgabe 10 vom Aufgabenblatt 3 weiter bzw. erweitern Sie jene.

## Aufgabe 25: Ungerade Zahlen

### Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Schleifen und einfache Arithmetik

### Aufgabenstellung

Welche Ausgabe liefert das folgende Programm? Ändern Sie die Initialwerte der Variablen i und j so ab, dass nur die ungeraden Zahlen von 13 bis 25 angezeigt werden?

```
#include<stdio.h>
int main()
{
    int i = 0;
    int j = 12;
    for(i; i < j-1; i += 2)
    {
        printf("%d ", i);
    }
    printf("\n");
```

```
    return 0;
}
```

## Aufgabe 26: Chemische Elemente

### Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Auswahlanweisung
- nicht-abweisende Schleife
- `scanf`-Besonderheiten

### Aufgabenstellung

a) Schreiben Sie unter Benutzung der Auswahlanweisung ein C-Programm, das vom Benutzer ein einzelnes Zeichen `c` einliest. Wenn es sich bei diesem Zeichen um ein chemisches Element handelt, soll dessen voller Name ausgegeben werden.
Beispiel: Eingabe: `'S'`, Ausgabe: `Schwefel`
*Hinweis:* Beim Einlesen eines einzelnen Zeichens mittels `scanf` sollten Whitespaces (so auch das Betätigen der Enter-Taste) übergangen werden. Dies kann durch ein Leerzeichen im Formatstring erreicht werden: `scanf(" %c", &c)`.

b) Ergänzen Sie Ihr Programm um die Ausgabe einer Fehlermeldung, wenn der Benutzer einen Buchstaben bzw. ein Zeichen eingegeben, für das kein chemisches Element existiert.

c) Erweitern Sie Ihr Programm so, dass das richtige chemische Element sowohl bei Eingabe eines Klein- oder Großbuchstaben ausgegeben wird. Beispiel: Sowohl im Falle des Eingabe von `'u'` als auch bei Eingabe von `'U'` soll `Uran` ausgegeben werden.

d) Betten Sie Ihren Code nun in eine nicht-abweisende Schleife ein, die solange wiederholt wird, bis der Benutzer als Zeichen die Null (`'0'`) eingegeben hat.

## Aufgabe 27: Vom Pseudocode zum Programmtext

### Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- Umwandeln von Pseudocode in C-Code
- `while`-Schleifen

### Aufgabenstellung

Der folgende Algorithmus zeichnet ein rechtwinkliges Dreieck auf dem Bildschirm. Übersetzen Sie diesen in C, kompilieren Sie Ihr Programm und führen Sie es aus. Testen Sie das resultierende Programm mit verschiedenen Eingaben für $a$.
*Hinweis:* Achten Sie auf die Einrückungen der einzelnen Zeilen, da diese nicht zufällig sind.

```
1. Erzeuge eine ganzzahlige Variable a
2. Erfrage den Werte von a
3. Wenn a größer als 0, tue das folgende:
  4. Erzeuge zwei Zählervariablen i und j
  5. Weise i den Wert 0 zu
  6. Solange i kleiner als a ist, tue das folgende:
    7. Weise j den Wert 0 zu
    8. Solange j kleiner/gleich i ist, tue das folgende:
      9. Gebe das Zeichen '*' aus
      10. Erhöhe den Wert von j um 1
    11. Gehe zu einer neuen Zeile (Zeilenvorschub)
    12. Erhöhe den Wert von i um 1
```

## Aufgabe 28: Zufallstext

### Inhalte und Sprachkonzepte der vorliegenden Aufgabe

- abweisende Schleife
- Zufallszahlengenerator

### Aufgabenstellung

Schreiben Sie ein C-Programm, das zufällige Zeichen auf dem Bildschirm ausgibt, ohne zu terminieren.

a) Realisieren Sie das geforderte Programm mittels einer abweisenden Endlosschleife. Jedes weitere auszugebende Zeichen soll zufällig aus dem Bereich der druckbaren ASCII-Zeichen ermittelt worden sein. Vergessen Sie nicht, den Zufallsgenerator adäquat zu initialisieren.
*Hinweis:* Die Ausgabe nach `stdout` erfolgt gepuffert, d.h. sie erfolgt erst dann, wenn eine größere Anzahl[2] Zeichen in den Ausgabepuffer der Konsole geschrieben worden sind. Sie können jedoch eine unmittelbare Ausgabe (auch eines einzelnen Zeichens) erzwingen, indem Sie nach einer `printf`-Anweisung den Ausgabepuffer mittels `fflush(stdout);` "leeren", d.h. auf den Bildschirm bringen. Testen Sie den Unterschied!

[2] Gelingt es Ihnen, während der Lösung dieser Aufgabe zu ermitteln, wie groß der Ausgabepuffer ist, d.h. wie viele Byte er umfasst?

b) Das in a) entwickelte Programm tätigt die Ausgaben auf die Konsole zu schnell, um sie mitzuverfolgen. Erweitern Sie Ihr Programm daher nun so, dass nach der Ausgabe jedes einzelnen Zeichens für eine bestimmte Zeit gewartet wird. Hierfür nutzen Sie die Funktion `usleep( w );` aus der Bibliothek `unistd.h`, die als Parameter `w` die Anzahl der Mikrosekunden erwartet, für die die Ausführung des Programms pausiert werden soll. Implementieren und testen Sie die folgenden beiden Varianten:

- Es soll für eine feste Zeitspanne von 20 Millisekunden (also 20.000 Mikrosekunden) nach jedem Zeichen gewartet werden.
- Es soll nach jedem Zeichen für eine zufällige Anzahl von Millisekunden aus dem Intervall $[10ms, 100ms]$ gewartet werden.