Funktionen

Einleitung

«Funktionen» dürften Euch bekannt sein aus der Mathematik. Zum Beispiel:

$$y = f(x)$$

Dabei wird der Wert x in die Funktion hineingegeben, damit werden irgendwelche Operationen durchgeführt, und das Resultat dem Wert y zugewiesen. Zum Beispiel:

$$y = 3x + 2$$

Ähnlich aber generischer sind Funktionen im Programmieren. Sie sind eine Sammlung von Instruktionen, die als Gesamtes eine ganz spezifische Aufgabe erfüllen. Wir kennen bereits einige Funktionen:

printf	gibt eine Ausgabe auf der Kommandozeile aus
main	Haupt- /Einstiegsfunktion für jedes C-Programm

Verständnis

Funktionen dienen dazu, Code zu strukturieren. Vor allem helfen Sie dabei, dass Code nicht wiederholt werden muss. Wir können eine Aufgabe generisch lösen und mit einem Funktionsaufruf (und entsprechenden Argumenten) den gleichen Code wiederverwenden. Funktionen sind demnach **benannte, unabhängige Code-Fragmente**.

Jede Funktion hat einen eineindeutigen Namen und gibt optional einen Wert zurück. Funktionen lassen sich aus anderen Funktion herausaufrufen (das kennen wir sowieso):

main → printf

Wir kennen den Compilerfehler «implicit function declaratio», der auftaucht, wenn wir z.B. printf verwenden, aber vergessen haben #include <stdio.h> aufzuführen. Deshalb unterscheiden wir **Deklarieren** und **Implementierung**. Zum Beispiel:

```c
#include <stdio.h>

// function declaration!
float ComputeCircleArea(float radius);


int main()
{
    float myRadius = 4.545;
    float area = ComputeCircleArea(myRadius);
    return 0;
}



// function implementation
float ComputeCircleArea(float radius){
    return radius*radius*3.1415926535;
```

Die Deklaration muss **zwingend** vor dem ersten Aufruf platziert werden! Alternative kann aber auch die ganze Implementierung vor dem ersten Aufruf stehen.

Aufgaben

1. Lese Dich schlau zum Thema «Rekursion». Danach erstelle eine Funktion «nchoosek», die den Binomialkoeffizienten berechnet:

   «n tief k» $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

   Hinweis: Teile das Problem auf. Programmiere zuerst eine Funktion, die n! (n-Faktorial) berechnen kann. Teste die Funktion in einem Programm.
   Berücksichtige Sonderfälle wie n = k = 0, n, k < 0

2. Ein zentrales Element in der Gestaltung von Funktionen sind die Konzepte von
   - pass-by-value Argument
   - pass-by-reference Argument

   Erarbeite anhand einer Onlinesuche den Unterschied und beschreibe den Unterschied im Wiki. Versuche bestehende Posts allenfalls zu ergänzen.

3. Mit diesem neu erarbeiteten Wissen, baue das Beispiel oben so um, dass die Fläche nicht als Rückgabewert, sondern als pass-by-reference Argument mitgeben werden kann.