Aufgabe 1: static

In dieser Übungsaufgabe sollen die Auswirkungen des Schlüsselwortes `static` erforscht werden. Dazu soll eine Bibliothek mit mathematischen Funktionen und ein passendes Anwendungsprogramm implementiert werden.

1. Die Bibliothek soll aus einer Header-Datei `mathemat.h` bestehen, die die öffentlichen Schnittstellen der mathematischen Funktionen bereitstellt und einer Implementierung der Funktionen der Schnittstelle und internen Funktionen in der Datei `mathemat.c`.

Die Bibliothek soll die Funktionen `float quadrat(float)` und `int xmod3(float)` ähnlich denen von Übungszettel 2, Aufgabe 2 bereitstellen. Als interne Funktion, die nicht von einer Anwendung aufgerufen werden kann, dient `int float2int(float)`. `float2int()` kann z.B. als Hilfsfunktion für `xmod3()` benutzt werden.

Implementieren Sie `mathemat.h` und `mathemat.c`. Nutzen Sie das Schlüsselwort `static` um Sichtbarkeiten einzuschränken. Testen Sie die Compilierfähigkeit Ihres Codes mit:

```
gcc -Wall -c mathemat.c.
```

Benutzen Sie folgende Vorlage für `mathemat.h`:

```
#ifndef INCLUDED_MATHEMAT_H
#define INCLUDED_MATHEMAT_H
// Your code goes here
#endif
```

Wir verzichten hier darauf, ein shared object zu erzeugen und gehen den etwas kürzeren Weg.

2. Implementieren Sie eine Anwendung `app.c`, die die Funktionen aus 1. testet. Compilieren und linken Sie Ihren Code mit `gcc -Wall app.c mathemat.c -lm -o app`.

Testen Sie Ihre Anwendung.

Was passiert, wenn Sie aus `app.c heraus float2int()` aufrufen wollen?

Fügen Sie den Prototypen von `float2int()` in `mathemat.h` hinzu. Wie verändert sich das Verhalten, wenn Sie nun versuchen, `float2int()` aus `app.c` heraus aufzurufen?

3. Sie möchten herausfinden, wie oft die Anwendung die mathematischen Funktionen aufruft. Nutzen Sie `static` nun, um nicht nur die `Sichtbarkeit` von Funktionen und globalen Variablen, sondern auch die `Lebensdauer` von lokalen Variablen zu manipulieren. Implementieren Sie dazu eine interne Hilfsfunktion `void counter(void)`. Diese Funktion soll eine lokale Variable als Zähler benutzen, die bei jedem Aufruf der Funktion inkrementiert wird und deren Wert mit `printf(3)` auf dem Terminal ausgegeben wird.

Ergänzen Sie den Aufruf der Funktion `counter()` in `quadrat()` bzw. `xmod3()` und testen Sie ihre Implementierung.