1 Bibliothek für Matrix-Operationen

Ziel dieser Aufgabe ist die Erstellung eines Programms, welches die Addition und Multiplikation zweier quadratischer 2D-Matrizen durchführt und das Ergebnis auf die Konsole ausgibt. Die Vorgabe-Datei `exercise_01.c` soll hierbei nicht verändert werden, lagern Sie die Implementation sämtlicher Funktionen in eine Library aus.

1.1 Implementation

Ihre Implementation soll die folgenden Funktionen unterstützen:

- Erzeugung des Speichers für eine Matrix auf dem Heap sowie Initialisierung mit Zufallszahlen zwischen 1 und 10. Sie können hierfür die Funktion `rand()` (vgl. `https://en.cppreference.com/w/c/numeric/random/rand`) verwenden.
- Addition zweier Matrizen
- Matrixprodukt zweier Matrizen
- Ausgabe der Berechnung auf die Konsole
- Freigabe von Matrix-Speicher

Die Ausgabe des vollständigen Programmes auf der Konsole könnte z.B. folgendermaßen aussehen:

```
5   3   7       2   3   6       7   6   13
4   5   9   +   1   4   9   =   5   9   18
3   1   1       8   3   2       11  4   3

5   3   7       2   3   6       69  48  71
4   5   9   *   1   4   9   =   85  59  87
3   1   1       8   3   2       15  16  29
```

Bauen Sie Ihre Implementation einmal als statische und einmal als shared Library und linken Sie sie in das Hauptprogramm. Notieren Sie die jeweils notwendigen Anweisungen, um die Libraries zu bauen.
Hinweis: Mit `typedef existierenderTyp neuerTyp` können Sie neue Typen definieren (wie z.B. `matrix` in der Vorlage). Das könnte z.B. dann so aussehen:

```
1 typedef struct Matrix {
2     ...
3 } Matrix;
4
5 typedef Matrix *matrix;
```

1.2 Zeitmessung

Messen Sie die Zeiten (z.B. mit Hilfe des `time`-Kommandos) für alle Bauschritte sowie die Ausführungszeit ihres Programmes für beide Varianten. Vergleichen Sie die Werte für die Verwendung von static und shared Libraries.

2 Geänderte Funktionalität

2.1 Startup

Bauen Sie das Programm `exercise_02_main.c` und führen Sie es aus. Im Terminal sollte nun eine rudimentäre Sinus-Kurve zu sehen sein.

2.2 Bildstörung

Die gegebene Implementation des Headers `exercise_02_check.h` verwendet die Funktion `sin` aus der C-Mathe-Library. Bedenken Sie hierbei, dass beim Bauen `-lm` notwendig ist, um die Math-Library zu linken, damit die Funktion `sin` zur Verfügung steht. Beeinflussen Sie die grafische Ausgabe ohne die vorgegebenen Dateien `exercise_02_main.c`, `exercise_02_check.h` oder `exercise_02_check.c` zu verändern oder das Ursprungs-Programm neu zu kompilieren.
Bauen Sie hierfür eine shared Library mit Ihrer eigenen Definition von `sin`, die ausgegebenen Werte müssen nichts mehr mit denen der tatsächlichen Sinus-Funktion gemein haben. Erzwingen Sie zur Laufzeit des Programms die Verwendung Ihrer eigenen *Sinus*-Funktion, indem Sie die passenden Umgebungsvariablen `LD_...` auf Ihre neue eigene shared Library zeigen lassen.
Statt der originalen Sinus-Funktion soll im Terminal dann eine andere Funktion ausgegeben werden; es bleibt Ihnen überlassen, welche Funktion sie darstellen wollen. Sie können z.B. ein Polynom, eine Dreieckskurve, Rechteckskurve, Sägezahnkurve oder etwas ganz anderes ausgeben lassen.

Material

Code-Vorlagen liegen dem beiliegenden Übungsmaterial bei. Da bei dieser Übung der Fokus auf dem manuellen Bauen liegt, werden dieses Mal keine Makefiles bereitgestellt.