Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
E. Betke, N. Hübbe, M. Kuhn, J. Lüttgau, J. Squar

Übungsblatt 10 zum Praktikum
C-Programmierung
im WiSe 2019/2020
Abgabe: 12.01.2020 (23:59)

1 Aufwärmrunde

Das Programm `10_modularitaet_00.c` entspricht einer möglichen Lösung vom sechsten Übungsblatt über *Debugging und Valgrind*. Das File ist relativ lang und unübersichtlich und möchte aufgeräumt werden. Lagern Sie alle Funktionen, die für den Umgang mit den Koordinaten nötig sind sowie die zugehörigen Structs in ein eigenes File aus. In `10_modularitaet_00.c` sollen nur die main- sowie die drei Test-Methoden sowie die notwendigen Header verbleiben.

2 Matrizen-Mathematik

Bewertete Aufgabe! (x/45)
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 `10_modularitaet_01.c` soll hierbei nicht verändert werden, lagern Sie die Implementation sämtlicher Funktionen in eine Library aus.

2.1 Implementation

Ihre Implementation soll die folgenden Funktionen unterstützen:

- Erzeugung des Speichers für eine Matrix sowie Initialisierung mit Zufallszahlen zwischen 1 und 10. Sie können hierfür die Funktion `rand()`[1] verwenden.
- Addition zweier Matrizen
- Matrizenprodukt 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:

```
1     5   3   7       2   3   6       7   6   13
2     4   5   9   +   1   4   9   =   5   9   18
3     3   1   1       8   3   2       11  4   3
4
5     5   3   7       2   3   6       69  48  71
6     4   5   9   *   1   4   9   =   85  59  87
7     3   1   1       8   3   2       15  16  29
```

[1] vgl. `https://en.cppreference.com/w/c/numeric/random/rand`

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.

2.2 Zeitmessung

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

3 Geänderte Funktionalität

3.1 Startup

Bauen Sie das Programm `10_modularitaet_02.c` und führen Sie es aus. Im Terminal sollte nun nun eine rudimentäre Sinus-Kurve zu sehen sein.

3.2 Bildstörung

Die gegebene Implementation des Headers `eingabeTrafo.h` verwendet die Funktion `sin` aus der C-Mathe-Library. Beeinflussen Sie die grafische Ausgabe ohne die vorgegebenen Dateien `10_modularitaet_02.c`, `eingabeTrafo.h` oder `eingabeTrafo.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 Umgebungsvariablen `LD_LIBRARY_PATH` und `LD_PRELOAD` auf Ihre neue 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.

4 Abgabe

Abzugeben ist ein gemäß den bekannten Richtlinien erstelltes und benanntes Archiv. Das enthaltene und gewohnt benannte Verzeichnis soll folgenden Inhalt haben:

- Alle Quellen, aus denen Ihr Programm besteht (10_modularitaet_01.c, 10_matrixOps.h, 10_matrixOps.c); gut dokumentiert(Kommentare im Code!)
- Beschreibung Ihrer Schritte zum Bau der Libraries sowie die Zeitmessung (10_-Evaluation.pdf)

Senden Sie Ihre Abgabe an cp-abgabe@wr.informatik.uni-hamburg.de .