# Übungsserie 3a

Programmierung mit C

*Damit für alle die gleichen Regeln gelten, müssen Ihre C-Programme in Standard-C (C11 oder C99) mit dem GCC oder Clang ohne Warnungen kompilieren. Vermeiden Sie insbesondere plattformspezifische Includes, z.B.* `windows.h`. *Schalten Sie alle Warnungen Ihres Compilers an um hilfreiche Informationen zu erhalten, Warnungen zu vermeiden und Fehler aufzuspüren. GCC und Clang unterstützen dies beispielsweise mit den Optionen:* `-Wall -Wextra -pedantic -std=c11`. *Bitte beachten Sie eventuell benötigte Bibliotheken zum Linken, z.B.* `-lm` *für das Linken gegen die Mathebibliothek. Eine ausgezeichnete Referenz zu C finden Sie unter: http://en.cppreference.com/w/c.*

*Bitte beachten Sie den Abgabezeitpunkt in Moodle. Serien per E-Mail oder verspätete Abgaben können nicht gewertet werden. Die Textaufgaben bitte per PDF mit den Programmieraufgaben zusammen in ein Zip-Archiv bündeln.*

**Aufgabe 1** (11 Punkte) Schreiben Sie die Implementierung `matrix.c` für das Matrix-Modul `matrix.h`. Laden Sie dazu die Datei `matrix.h` aus Moodle URL: https://moodle2.uni-leipzig.de/mod/resource/view.php?id=522151. Verändern Sie diesen Header nicht, sondern implementieren Sie alle Prototypen.[1] Achten Sie insbesondere auf Absicherungen der Dereferenzierung zur Vermeidung von Speicherzugriffsfehlern. Testen Sie Ihre Modulimplementation. Schreiben Sie dazu ein kleines Programm `matrixMain.c`, welche das Matrix-Modul `matrix.h` verwendet.

**Aufgabe 2** (7 Punkte) Gegeben sei die Datenstruktur Punkt wie folgt:

```
typedef struct { double x; double y;} Punkt;
```

Schreiben Sie ein Programm `punkt.c` welches ein Array von Punkten mit Hilfe der Standardfunktion `qsort`[2] anhand der euklidischen Norm eines Punktes sortiert. Gehen Sie dabei wie folgt vor und implementieren Sie folgende Funktionen:

- `double norm( Punkt const * const p );` welche die Euklidische Norm eines Punktes berechnet ($\sqrt{x^2 + y^2}$)
- `Punkt* randomPunkte( size_t len );` um ein Array von Punkten gegebener Länge zu erzeugen. Die x und y Komponenten der Punkte sollen dabei zwischen -100.0 und +100.0 liegen.
- `void printPunkte( Punkt const * const pts, size_t len );` um ein Array von Punkten und Ihre Norm auszugegeben. Nutzen Sie dazu folgenden Formatstring: `"(%.3g, %.3g) => %.3g | "`. Dabei sind die ersten beiden Argumente die x und y Komponenten der Punkte und das letzte Argument ist die Norm. Beispiel:

[1] Matrixmultiplikation vgl. https://de.wikipedia.org/wiki/Matrizenmultiplikation
[2] Vgl. http://en.cppreference.com/w/c/algorithm/qsort

```
(-75.1, 16.1) => 76.8 | (-7.66, -78.1) => 78.5 | (-68.2, 9.71) => 68.9
|
```

- `int comparePunkte(const void* p1, const void* p2);` als Vergleichsfunktion für die Sortierfunktion `qsort`. Diese soll -1 zurückgeben für den Fall dass die Norm des ersten Punktes (p1) kleiner ist als die Norm des zweiten Punktes (p2), +1 für den Fall dass die Norm des ersten Punktes größer ist als die des zweiten, und 0 falls sie die gleiche Norm haben. Dabei sollen insbesondere die Punkte nicht verändert werden.

- `int main(int argc, char const *argv[]);` um Ihr Programm zu testen und `qsort` auf ein zufällig erzeugtes Array anzuwenden. Die Länge des Arrays soll per Kommandozeile übergeben werden. Beispielaufruf:
  `./PunktProgramm.out 10`

  Implementieren Sie ein Errorhandling für falsche oder fehlende Parameter in der Kommandozeile. Geben Sie dazu vor dem Sortieren das Array mit `printPunkte` auf die Konsole aus, und ebenso anschließend nach dem Aufruf von `qsort`. Die obige Sequenz sortiert wäre dann:

  ```
  (-68.2, 9.71) => 68.9 | (-75.1, 16.1) => 76.8 | (-7.66, -78.1) => 78.5
  |
  ```