

## 1 2D Arrays

Implementieren Sie eine Funktion, die einen 2D Array als Matrix auf der Konsole ausgeben kann. Diese Funktion soll mit folgendem Aufruf nutzbar sein:

```
1 int main() {
2     const size_t width = 4, height = 3;
3     double data[height][width];
4     for(size_t y = 0; y < height; y++) {
5         for(size_t x = 0; x < width; x++) {
6             data[y][x] = 0;
7             if(y == 1 && x == 1) data[y][x] = 4;
8             if(y == 1 && x == 2) data[y][x] = 2;
9         }
10    }
11
12    print2dArray(height, width, data);
13 }
```

Erwartete Ausgabe:

```
0.000000 0.000000 0.000000 0.000000
0.000000 4.000000 2.000000 0.000000
0.000000 0.000000 0.000000 0.000000
```

## 2 Callbacks

Implementieren Sie eine Funktion, der Sie eine Callback-Funktion übergeben können. Die Callback-Funktion soll die Signatur **double** foo(**double** x, **double** y) haben. Außer dem Callback-Parameter soll die Funktion ein 2D Array übergeben bekommen. Ziel ist es, die Rückgabewerte der Callback-Funktion in den übergebenen Array zu schreiben. Die Funktion soll mit folgendem Code aufrufbar sein:

```
1 #include <assert.h>
2 #include <stdlib.h>
3
4 double foo(double x, double y) {
5     return x*x + y;
6 }
7
8 int main() {
9     const size_t height = 2, width = 3;
10    double data[height][width];
11    const double x_min = -1, x_max = 1;
12    const double y_min = 1, y_max = 3;
13
14    fillMatrixWithResults(height, y_min, y_max,
```

```
15         width, x_min, x_max,
16         data, &foo);
17
18     assert(data[0][0] == 2);
19     assert(data[0][1] == 1);
20     assert(data[0][2] == 2);
21     assert(data[1][0] == 4);
22     assert(data[1][1] == 3);
23     assert(data[1][2] == 4);
24 }
```

### 3 Integration

Schreiben Sie ein Programm, das die beiden Funktionen aus den vorigen Aufgaben nutzt um die Werte folgender Funktionen auf die Konsole auszugeben:

- $f(x, y) = x \cdot y$  im Bereich  $x, y \in [-1, 1]$ , Schrittweite  $\frac{1}{2}$
- $g(x, y) = \sin x \cdot \sin y$  im Bereich  $x, y \in [0, \pi]$ , Schrittweite  $\frac{\pi}{4}$
- $h(x, y) = \max(x, y)$  im Bereich  $x, y \in [0, 10]$ , Schrittweite 1

Hinweis: Um in C die Funktion `sin()` verwenden zu können, muss der Header `math.h` inkludiert werden und das Programm mit dem Flag `-lm` gelinkt werden. Dabei muss `-lm` nach den Objektdateien/ `.c` Dateien angegeben werden. Eine Funktion `max()` gibt es nicht in der Standardbibliothek, die müsst Ihr also selber programmieren.