Das Schönste, was wir erleben können, ist das Geheimnisvolle. Es ist das Grundgefühl, das an der Wiege von wahrer Kunst und Wissenschaft steht.
Albert Einstein

UNIVERSITÄT DES SAARLANDES

FR Mathematik
Andreas Buchheit

6. Übung zur Vorlesung

Programmierung

im Sommersemester 2019

Abgabe: Mittwoch, den 29.05.2019 bis spätestens 12 Uhr.

Aufgabe 6.1. (6 Punkte) Array Arithmetik
Im Folgenden sollen Sie grundlegende mathematische Operationen für ein- und zweidimensionale Arrays des Typs `double` implementieren.
Implementieren Sie Prozeduren die

- die Multiplikation eines Arrays (eine Prozedur für 1D, und eine weitere für 2D) mit einem Skalar,
- die komponentenweise Addition zweier Arrays (1D + 2D) ,
- die Matrix-Vektor Multiplikation,
- die Matrix-Matrix Multiplikation,

berechnen. Wählen Sie ein sinnvolles Interface! Hier ein Beispiel für den Protopypen der Matrix-Matrix Multiplikation:

```
void matrix_matrix_mult(size_t rows1,
                        size_t cols1,
                        size_t cols2
                        double mat1[rows1][cols1],
                        double mat2[cols1][cols2],
                        double to[rows1][cols2],
                        );
```

Hierbei wird das Matrix Produkt in das 2D Array `to` geschrieben. Testen Sie die Prozeduren anhand einfacher Matrizen und Vektoren.

Bitte wenden!

Aufgabe 6.2. (8 Punkte) Sierpinski-Dreieck

In dieser Aufgabe sollen Sie ein *Sierpinski-Dreieck*, wie in der Abbildung dargestellt, zeichnen. Ein solches Sierpinski-Dreieck entsteht durch wiederholte Aufteilung eines Dreiecks in vier kongruente Dreiecke oder durch ein iteriertes Funktionensystem, welches im Folgenden vorgestellt wird.

Das Sierpinski-Dreieck wird auf dem 2-dimensionalen Intervall $[0,1] \times [0,1]$ dargestellt. Zunächst wird *zufällig* ein Punkt $p_0 \in [0,1] \times [0,1] \subset \mathbb{R}^2$ gewählt. Die schwarzen Punkte p_k für $k = 1, 2, \ldots$ ergeben sich nun sukzessiv, indem *zufällig* eine der Funktionen

```
void sierpinski_0(double p[2]) {
  p[0] = p[0]/2;
  p[1] = p[1]/2;
}

void sierpinski_1(double p[2]) {
  p[0] = (p[0]+1)/2;
  p[1] = p[1]/2;
}

void sierpinski_2(double p[2]) {
  p[0] = p[0]/2;
  p[1] = (p[1]+1)/2;
}
```

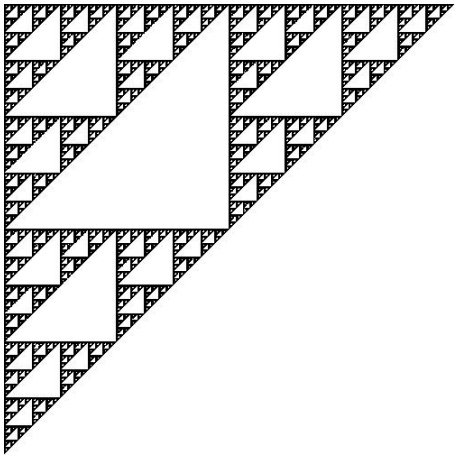

Abbildung: Sierpinski-Dreieck

ausgewählt wird und diese auf den vorherigen Punkt p_{k-1} angewendet wird. Um also p_1 zu bestimmen, wählen wir *zufällig* eine dieser Funktionen und wenden sie auf p_0 an. Um p_2 zu bestimmen wenden wir eine *zufällig* gewählte Funktion auf p_1 an. Um p_3 zu bestimmen ... Dieser Vorgang soll mehrere Tausend mal wiederholt werden.

Nun zur Implementierung. Schreiben Sie ein Programm, das zunächst die Anzahl der gewünschten Iterationen (der schwarzen Punkte) einliest. Das Intervall $[0,1] \times [0,1]$ soll mit einem 2-dimensionalen Feld vom Typ `unsigned char` der Größe 500 mal 500 dargestellt werden. Verwenden Sie für die Größe des Feldes zwei *Präprozessorkonstanten* `HEIGHT` und `WIDTH`, damit sie später leicht verändert werden können. Der Feldeintrag 255 bedeutet weiß und der Eintrag 0 schwarz. Initialisieren Sie das Feld mit „weiß“. Berechnen Sie nun innerhalb einer Funktion die schwarzen Punkte gemäß der oben erläuterten Regel und setzen Sie den entsprechenden Feldeintrag auf „schwarz“. Um einen Punkt p in das Feld einzutragen müssen Sie ihn zunächst auf den Indexbereich des Feldes skalieren. Dies geschieht mit:

```
int i = (int) ( p[0] * (HEIGHT - 1) );
int j = (int) ( p[1] * (WIDTH - 1) );
```

wobei `i` der Zeilenindex und `j` der Spaltenindex des Feldes ist.

Haben Sie alle Punkte eingetragen, so speichern Sie Ihr Feld als Bilddatei im Format PGM (Portable Graymap) ab. Hierzu rufen Sie einfach die folgende Funktion mit dem Namen des Bildes, der Höhe sowie der Breite des Feldes und Ihrem Feld auf

```
void write_pgm(const char *filename,
               size_t height,
               size_t width,
               unsigned char array[][width]
               );
```

die in der auf der Website hochgestellten Datei `pgm_utils.h` implementiert wird. Das erzeugte Bild lässt sich unter anderem mit Gimp betrachten.