Institut für Informatik Sommersemester 2019
Albert-Ludwigs-Universität Freiburg
Prof. Dr. Peter Thiemann
Dr. Philipp Klaus Krause

8. Programmieren in C

Abgabe bis 2. Juli, 10:15 GMT+2.

Aufgabe 0:

Schreibe ein Programm `gles_life` auf Grundlage von Aufgabe 2, Blatt 7 wie folgt!

Bei Druck der Taste 'h' soll die Ansicht um den Punkt $(0, 0)$ zentriert werden, so dass auch das grüne Quadrat noch voll sichtbar ist. Bei Druck von 'a' soll auf die Mitte der beiden Quadrate zentriert werden, so dass beide Quadrate voll sichtbar sind. Nach je einem Druck von '+' sollen die Quadrate doppelt so groß wie zuvor erscheinen, bei '-' halb so groß. Nach je einem Druck auf '→' sollen die Quadrate weiter links erscheinen (entsprechend auch für die anderen Pfeiltasten). Ob für '+' und '-' die Tasten auf dem Ziffernblock, oder die anderen, oder beide, verwendet werden, bleibt euch überlassen.

Aufgabe 1:

Schreibe ein Programm `otp`, das Verschlüsselung mittels eines One-Time-Pad implementiert. Das Programm wird mit drei Kommandozeilenargumenten aufgerufen; das erste gibt den Namen der zu ver- oder entschlüsselnden Datei an. Das zweite gibt die Schlüsseldatei an. Das dritte gibt den Namen der Ausgabedatei an. Als Verschlüsselung wird bitweises XOR (Operator `^` auf den einzelnen Bytes verwendet). Falls die Schlüsseldatei kürzer als die Eingabe ist, wird nur soviel ver- oder entschlüsselt, wie mit der Schlüsseldatei möglich, und keine weitere Ausgabe mehr in die Ausgabedatei geschrieben. In diesem Fall wird auch eine Fehlermeldung ausgegeben. Falls es bei der Dateiein- oder -ausgabe zu Fehlern kommt, ist ebenfalls eine Fehlermeldung auszugeben. Das Programm soll keinen dynamisch allozierten Speicher verwenden und keine obere Schranke an die Dateigröße aufweisen!

Aufgabe 2:

Schreibe ein Programm `gles_ulam`, auf Grundlage von Aufgabe 2, Blatt 7 unter Verwendung von Aufgabe 0, Blatt 3 und Aufgabe 0, Blatt 7 wie folgt!

Um jedes $(x, y) \in \{-2, 1, 0, 1, 2\}^2$, so dass $\mathfrak{U}(x, y)$ nicht prim ist, wird ein weißes Quadrat dargestellt (die oberste und unterste Reihe werden allerdings im Fenster nicht sichtbar sein).

Aufgabe 3:

Erweitere die Implementierung aus Aufgabe 1, Blatt 7 wie folgt! Das Interface erhält die zusätzlichen Funktionen:

```
// Kleinstes z, so dass ein Wert bei (x, z) oder (z, y) ungleich 0 ist.
// Gibt INT_MAX zurück, falls es keinen solchen Wert gibt.
int intplane_get_min(const intplane_t *p);
// Größtes z, so dass ein Wert bei (x, z) oder (z, y) ungleich 0 ist.
// Gibt INT_MIN zurück, falls es keinen solchen Wert gibt.
int intplane_get_max(const intplane_t *p);
```