Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TU Kaiserslautern
Fachbereich Informatik

Übungsblatt 11: Programmieren in C (WS 2019/20)

Abgabe: Montag, 03.02.20, 12:00

Geometrische Figuren

Abgabe: `geom.c`

Scalable Vector Graphics (SVG) ist ein Bildformat zur Beschreibung von zweidimensionalen Vektorgraphiken. Die Beschreibung beginnt mit dem `<svg>`-Tag, bei dem die Bilgröße (`width` und `height`), die SVG-Version sowie das Rendering-Schema anzugeben sind. Dann folgen die Tags mit den grafischen Elementen (z.B. Rechtecke); am Ende muss der Abschlusstag `</svg>` gesetzt werden.

Hier ein Beispiel:

```
<svg width="1000" height="1000" version="2.0" xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="200" height="100" fill="yellow"/>
<rect x="200" y="200" width="200" height="400" fill="green"/>
<rect x="250" y="250" width="100" height="100" fill="red"/>
</svg>
```

Der Punkt `x=0, y=0` beschreibt dabei die linke obere(!) Ecke des Bildes.

Sie können SVG-Dateien in Ihrem Webbrowser öffnen, um sich die entsprechende Grafik anzusehen. Im folgenden werden wir Grafiken im SVG-Format analysieren, optimieren und erzeugen. Laden Sie sich dazu die Vorlage `geom.c` sowie die Testdatei `demo.svg` von der Vorlesungsseite herunter.

Aufgabe 1 *Modellieren von geometrischen Figuren - Rechtecke (10 Punkte)*

Erstellen Sie eine C-Struktur `Rectangle` zur Modellierung von Rechtecken. Diese soll Informationen zum Referenzpunkt (`x`, `y`), Höhe (`height`), Breite (`width`) sowie Farbe (`color`, als String mit max. 10 Zeichen) umfassen.

Schreiben/ergänzen Sie nun das Programm zum Einlesen einer SVG-Grafik mit Rechtecken. Das Programm soll eine SVG-Grafik ausgegeben, die alle Rechtecke in der Komplementärfarbe anzeigt:

Farbe	Komplementärfarbe
red	green
blue	orange
yellow	purple

Beachten Sie dabei folgende Hinweise:

- Überlegen Sie sich eine geeigenete Aufteilung in Funktionen. Bitte kommentieren Sie Ihren Code außerdem sinnvoll! Bei Abgaben, die nicht strukturiert oder unkommentiert sind, werden wir ggf. Punkte abziehen.
- Sie können für dieses Übungsblatt annehmen, dass die eingelesenen Grafiken max. 10 Rechtecke umfassen. Legen Sie dazu ein geeignetes Array an.
- Verwenden Sie die bereits vorgegebene Funktion `void parse_rect(char *str, struct Rectangle *r)`, die die Elemente des Rechtecks aus dem String `str` extrahiert und in der mit `r` referenzierten Struktur einträgt.
- Verwenden Sie die Funktion `fgets`, die Sie auf den vorherigen Übungsblättern kennengelernt haben. Die Anzahl an Charactern pro Zeile der SVG-Datei beträgt max. 100.
- Sie können davon ausgehen, dass nur die Farben in der folgenden Tabelle verwendet werden und die Grafik max. Breite `1000` und Höhe `1000` hat (wie im obigen Beispiel).

- Zum Einlesen und Ausgeben verwenden Sie bitte folgende Anweisung auf der **Kommandozeile** (DevC++ unterstützt dies nicht direkt) unter Windows:

```
geom.exe < input.svg > output.svg
```

  Unter Linux (Terminalrechner) bzw. MacOS:

```
clang geom.c -o geom
./geom < input.svg > output.svg
```

  Sie können zur Kontrolle dann die Dateien `input.svg` und `output.svg` in einem Webbrowser öffnen.

Aufgabe 2 *Bildgröße minimieren (5 Punkte)*

Ergänzen Sie Ihr Programm, so dass die Breite und Höhe der Grafik nicht mehr konstant `1000` ist, sondern möglichst klein gewählt sind und dabei alle Rechtecke vollständig dargestellt werden können.

Aufgabe 3 *Duplikate eliminieren (5 Punkte)*

Ergänzen Sie Ihr Programm, so dass identische Rechtecke nur einmal in der Ausgabe aufgelistet werden. Achten Sie dabei auf die Reihenfolge, in der die Rechtecke grafisch ausgegeben werden!

Verkette Liste

Abgabe: `linkedlist.c`

Laden Sie sich die Datei `linkedlist.c` herunter, welche die Implementierung von einfach verketteten Listen aus der Vorlesung enthält. In dieser Aufgabe sollen Sie diese Liste um weitere Funktionen erweitern.

Aufgabe 4 *Test auf Sortiertheit (5 Punkte)*

Schreiben Sie eine Funktion `bool list_is_sorted(linked_list_t *ll)`, welche prüft ob die in der Liste `ll` gespeicherten Werte aufsteigend sortiert sind.

Aufgabe 5 *Duplikate (5 Punkte)*

Schreiben Sie eine Funktion `bool list_has_duplicates(linked_list_t *ll)`, welche prüft, ob in der Liste `ll` Werte existieren, die mehrmals vorkommen.

Aufgabe 6 *Einfügen (5 Punkte)*

Schreiben Sie eine Funktion `void list_add_before(linked_list_t *ll, int x, int y)`, welche einen neuen Eintrag mit Wert `x` direkt vor dem ersten Eintrag mit Wert `y` in die Liste einfügt. Wenn `y` nicht in der Liste enthalten ist soll `x` am Ende der Liste eingefügt werden.

Aufgabe 7 *Löschen von Elementen* ***[freiwillig, Achtung: schwierig]***

Schreiben Sie eine Funktion `int list_remove(linked_list_t *ll, int value)`, welche alle Vorkommen von `value` aus der Liste `ll` entfernt und zurückgibt, wie viele Elemente entfernt wurden. Die Liste soll dabei von der Funktion nur einmal durchlaufen werden.