Praktikum 2: Linien clippen

Kopieren Sie den Code dieses Praktikums in den des letzten Praktikums. In der Klasse `LineRasterizerMeyer` wird die Klasse `LineClipping` durch den Aufruf

```
1  ClipResult clip = LineClipping.clip(a, b, 0, 0, w-1, h-1);
```

bereits korrekt benutzt. Stellen Sie sicher, dass in Ihrer Klasse `LineRasterizer` dies auch geschieht.

In dieser Aufgabe solle das Verfahren von Cohen-Sutherland zum Clipping von Linien implementiert werden. Um zu sehen, ob das Clipping im Inneren klappt können Sie ja den Clip bereich in den Bildschirm legen mit

```
1  int d = 40;
2  ClipResult clip = LineClipping.clip(a, b, 0+d, 0+d, w-1-d, h-1-d);
```

Aufgabe 1 Ein-Zeiler implementieren

a) Implementieren Sie die Methoden `intersectX` und `intersectY` der `ClipResult`-Klasse entsprechend ihrer Kommentare! Leiten Sie dazu auf Papier die die Funktionsgleichung $f(x)$ der Line von $\vec{a}$ nach $\vec{b}$ her und lösen sie nach x bzw. y auf um die Schnittpunkte entsprechend zu bestimmen. Runden Sie Ihre Ergebnisse auf Integer!

b) Outcodes und ein Code für innen werden in der Klasse LineClipping als Integer definiert:

```
1  //0B ist das Prefix ist für Binärzahlen
2  public static final int left   = 0B1000;
3  public static final int right  = 0B0100;
4  public static final int top    = 0B0010;
5  public static final int bottom = 0B0001;
6  public static final int inside = 0B0000;
```

Nutzen Sie geeignete Bit-Operationen um den Outcode eines Punkts $\vec{p}$ bezüglich des Rechteckes $[x_{\text{min}}..x_{\text{max}}] \times [y_{\text{min}}..y_{\text{max}}]$ zu bestimmen! Implementieren Sie dazu die Methode `LineClipping.outcode`!

c) Ebenfalls mit Bit-Operationen, aber auch mit logischen Operationen, können Sie nun erkennen, ob die Outcodes `outA`, `outB` zweier Punkte a und b ein trivial Reject bzw. ein trivial Accept darstellen. Implementieren Sie diese Prädikate in den Methoden `trivialAccept` und `trivialReject`!

d) Implementieren Sie nun die Methoden `isLeftOut`, `isRightOut`, `isTopOut` und `isBottomOut` gemäß den Kommentaren!

Aufgabe 2 Cohen-Sutherland Clipping

Clippen Sie nun die Linien mit dem Cohen-Sutherland Algorithmus in der Methode `clip`. Sie können folgenden Pseudo-Code als Hilfe benutzen:

1. Berechne Outcodes von $\vec{a}$ und $\vec{b}$
2. Solange kein Trivial Accept oder keine trivial Reject vorliegt:
    a. Wähle genau einen Punkt der außerhalb liegt
    b. Wähle genau eine Achse für die dieser Punkt außerhalb liegt
    c. Schneide die Linie gegen diese Achse
    d. Aktualisiere die Koordinaten des gewählten Punktes
    e. Aktualisiere Outcode des gewählten Punktes
3. Gib Line zurück.

Wenn alles klappt, dann können Sie in `A1Lines.pde` zum Testen diese Methode hinzufügen und aufrufen. Diese erzeugt Linien für die ein Punkt außerhalb und der andere innerhalb liegt!

```
1 void test_animation_Clip()
2 {
3   float alpha = millis()/1000.0;
4   drawTestLine(alpha, 1.9f, #ff00ff00);
5 }
```