Übungsblatt 10 (Projektaufgabe)

(14. Dezember 2009)

Dieses ist die Semsteraufgabe, für die es maximal 20 Punkten gibt. Die Lösung ist in schriftlicher und datenelektronischer Form beim Übungsleiter abzugeben. Letzter Abgabetermin ist der 18. Januar. Das Deckblatt der schriftlichen Lösung enthält bitte Name und Matrikelnummer und eine unterschriebene Erklärung, dass die Lösung selbständig angefertig wurde. Die Lösungen sind nicht in Gruppenarbeit anzufertigen.

Benotet wird nach:

- Funktionalität
- Code Qualität
- Dokumentation
- Optik der generierten Bilddateien

Zusätzlich müssen Sie darauf vorbereitet sein, eventuell in der letzten Woche Fragen zu Ihrer Lösung beantworten zu können.

Aufgabe 1 Schreiben Sie eine kleine Bibliothek, mit deren Hilfe Sie Diagramme erzeugen und in eine Bitmapdatei speichern können. Benutzen Sie hierzu die Blibliothek `BMP.h` von der Webseite.

a) Schreiben Sie eine Prozedur
`void background(Bmp* this,Color c)`
Sie soll das ganze Bild mit einer Hintergrundfarbe ausfüllen.

b) Schreiben Sie eine Prozedur
`void fillRect(Bmp* this,Color c,int x,int y,int w,int h)`
Sie soll ein farbiges Rechteck mit der linken unteren Ecke `(x,y)`, der Weite `w` und der Höhe `h` in die Bitmapdatei zeichnen.

c) Schreiben Sie eine Prozedur
`void drawLine(Bmp* this,Color c,int fromX,int fromY,int toX,int toY)`
Sie soll eine farbige Linie mit dem Startpunkt `(fromX,fromY)` und dem Endpunkt `(toX,toY)` in die Bitmapdatei zeichnen.

d) Schreiben Sie eine Prozedur
`void fillCircle(Bmp* this,Color c,int x,int y,int radius)`
Sie soll einen farbigen Kreis mit dem Mittelpunkt `(x,y)` und dem Radius `radius` in die Bitmapdatei zeichnen.

e) Schreiben Sie eine Funktion
`Bmp* barChart(double values[],int valueLength)`,
die für die übergebenen Daten ein Säulendiagramm in einer neuen Bitmapdatei erzeugt. Jeder Wert der übergebenen Reihung soll dabei eine Säule in einer eigenen Farbe bekommen, deren Höhe proportional zur Höhe des Wert ist.

f) Schreiben Sie eine Prozedur
`Bmp* pieChart(double values[],int valueLength)`,
die für die übergebenen Daten ein Kuchendiagramm in einer neuen erzeugt. Jeder Wert der übergebenen Reihung soll dabei eine Kreissegment in einer eigenen Farbe bekommen, dessen Winkel proportional zur Höhe des Wert ist.

g) Erzeugen Sie mit den letzten beiden Prozeduren Diagramme, die das Wahlergebnis der letzten Bundestagswahl darstellen.

Aufgabe 2 (Wahlweise alternativ zu Aufgabe 3)
In dieser Aufgabe sollen Sie Funktionsgraphen in einer Bitmapdatei darstellen.

a) Schreiben Sie eine Funktion
`Bmp* drawGraph(double (f)(double),double minX,double maxX)`,
die den Funktionsgraphen der Funktion `f` im Wertebereich von `minX` bis `maxX` in eine Bitmapdatei zeichnet. Die Datei soll hierfür neu erzeugt werden. Ihre Größe soll dabei von den darzustellenden Graphen abhängen. In der erzeugten Datei soll auch das Koordinatensystem mit eingezeichnet sein.

b) Die Ableitung einer Funktion (auch Differentialquotient genannt) ist definiert durch $\frac{f(x_0+\Delta x)-f(x_0)}{(x_0+\Delta x)-x_0}$ für Δx gegen 0. Sie können also mit einem sehr kleinen Wert für Δx die Ableitung in einem bestimmten Punkt annähern.

Schreiben Sie eine Funktion:
`double ab(double (f)(double),double x)`,
die für eine Funktion den Wert der Ableitung an einer bestimmten Position `x` berechnet.

c) Schreiben Sie eine Funktion
`double abN(double (f)(double),double x,unsigned int n)`,
die den Wert der n-ten Ableitungsfunktion an einer bestimmten Stelle x für eine Funktion berechnet.

Benutzen Sie bei der Umsetzung die gcc-Erweiterung der Sprache C, die lokale (nested) Funktionsdefinitionen zulässt.
(siehe z.B.: `http://en.wikipedia.org/wiki/Nested_function`).

Realisieren Sie die Funktion rekursiv. Für $n = 0$ ist das Ergebnis `f(x)` ansonsten `abN(f' ,x,n-1)`, mit $f'(x) = ab(f, x)$.

d) Schreiben Sie eine Funktion

```
Bmp* drawGraphWithDiffs
       (double (f)(double), double minX, double maxX
       ,unsigned int numberOfDiffs),
```

die den Funktionsgraphen der Funktion `f` im Wertebereich von `minX` bis `maxX` in eine neu erzeugte Bitmapdatei zeichnet. Zusätzlich sollen die Graphen der Ableitungen in die gleiche Datei in je einer unterschiedlichen Farbe gezeichnet sein.

Es sollen also für `numberOfDiffs` mit dem Wert 2 insgesamt 3 Graphen in eine Datei gezeichnet werden: die Graphen von f und der ersten und zweiten Ableitung von f.

e) Erzeugen Sie mehrere Beispieldateien mit Funktionsgraphen und deren Ableitungen. Hierbei von einigen Polynomen und einer trigeometrischen Funktion.

Aufgabe 3 (Wahlweise alternativ zu Aufgabe 2)
Einer Turtlegraphik ist eine spezielle Art zu zeichnen. Man hat eine »Schildkröte«, die entweder gerade aus laufen kann, oder sich drehen kann. Wenn die Schildkröte gerade aus läuft, dann kann eine Linie gezeichnet werden. (Mehr `http://de.wikipedia.org/wiki/Turtle-Grafik`).

a) Implementieren Sie eine Datenstruktur für die Turtle-Graphik. Eine Turtle-Graphik bestehe dabei aus den aktuellen Werten der Schildkröte und aus der Bitmapgraphik, auf der sich die Schildkröte bewegt. Die Schildkröte soll sich um einen bestimmten Winkel drehen können, oder in die Richtung, in die sie blickt laufen. Die Schildkröte hat zusätzlich eine Farbe eingestellt.

b) Malen Sie mit der Turtle-Graphik ein Treppenmuster in eine Bitmapdatei..

c) Malen Sie mit der Turtle-Graphik ein Mäander wie auf
`(http://upload.wikimedia.org/wikipedia/de/c/cf/M%E4ander.png)`
dargestellt (ohne die einfachen Linien oben und unten).

d) Zeichnen Sie einen Kreis mit der Turtle-Graphik. Der Kreis bestehe aus sehr kleinen Sekanten.

e) Malen Sie eine Spirale mit der Turtle-Graphik. Dazu wird eine langsam größer werdende Linie gezeichnet, mit immer den gleichen (kleinen) Winkel.

Aufgabe 4 In dieser Aufgabe greifen wir die Reime der letzten zwei Übungsblätter noch einmal auf.

a) Entwerfen Sie eine Datenstruktur `RhymePair`, die zwei sich reimende Wörter speichern kann. Schreiben Sie entsprechende Konstruktor- und Destruktorfunktionen.

b) Entwickeln Sie eine Datenstruktur `RhymePairList` für einfach verkettete Listen, die Zeiger auf Reimpaarobjekte speichern kann. Schreiben Sie entsprechende Konstruktor- und Destruktorfunktionen.

c) Schreiben Sie eine Funktion:
`RhymePairList* fromWordList(WordList* ws)`, die aus einem WordList-Objekt die Liste aller Reimpaareerzeugt, die in dieser Liste enthalten sind.

d) Schreiben Sie eine Funktion
`boolean startsWithSameConsonants(string s1,string s2)`,
die testet, ob zwei Wörter mit denselben Konsonanten starten, d.h.: dass sie bis zum ersten Vokal identisch sind.

e) Schreiben Sie eine Funktion
`RhymePair* spoonerism(RhymePairList* this,RhymePair* rhymeWords)`,
die zu einem Paar sich reimender Wörter ein zweites Reimpaar zurückgibt, so dass beide Reimpaare zusammen einen Schüttelreim (englisch: spoonerism) ergeben. Das zweite Reimpaar muss hierzu mit denselben Konsonanten beginnen wie das erste Reimpaar. Z.B.: für das Reimpaar Mund, Hund ist ein passendes Reimpaar zur Bildung eines Schüttelreims: Maus, Haus, der Schüttelreim entsteht dann überkreuzt: Haus-Mund, Maus-Hund (was in diesem Fall wenig Semantik hat).

f) Benutzen Sie Ihr Programm, um einen interessanten Schüttelreim zu finden. Lesen Sie hierzu eine beliebige Textdatei ein, lassen diese in Wörter zerlegen, erzeugen die Liste aller Reimpaare, nehmen ein Reimpaar aus der Liste und suchen in der Liste ein entsprechendes Schüttelreimpaar.
Ein Javaapplet, das nach diesem Prinzip funktioniert, können Sie, sofern Java in Ihren Browser aktiviert ist und genügend Speicherplatz zur Verfügung hat, auf `http://www.sveneric.de/shaker.html` ausprobieren.

g) Verarbeiten Sie das in der letzten Aufgabe gefundene Schüttelreimpaar in zwei Verszeilen. Der schönste Zweizeiler wird mit einem Buchgeschenk premiert.