Programmieren für Physiker

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. A. Mildenberger
http://comp.physik.kit.edu

SS 2017 – Blatt 06
Bearbeitungszeitraum: bis 07. Juni 2017

Aufgabe 15: ASCII-Bilder dekodieren **Pflichtaufgabe**

In den Dateien `a15-bildX.dat`, `X`= 1 . . . 4, die sich im Verzeichnis `/home/ck17/daten` und auf der Webseite befinden, ist jeweils ein Bild kodiert. Dabei wird folgendes Codierungsformat verwendet: Die Textdateien enthalten zunächst in zwei Zahlen die X- und Y-Größe des Bildes, danach folgen bis zum Dateiende Wertepaare. Die erste Zahl jedes Wertepaares steht für einen Zeichencode und die zweite Zahl gibt die Anzahl an, wie oft dieses Zeichen hintereinander auszugeben ist. Die Bilder sind Zeile für Zeile von oben nach unten gespeichert.
Für den Zeichencode gilt in dieser Aufgabe folgende fest vorgegebene Tabelle:

Zeichencode	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Zeichen	`.`	`,`	`~`	`=`	`+`	`:`	`?`	`$`	`7`	`8`	`D`	`I`	`M`	`N`	`O`	`Z`

Beispiel: Steht in der Datei `3 5`, so sind fünf Gleichheitszeichen auszugeben.

Schreiben Sie ein Programm, das eine Datei, die nach dem angegebenen Schema codiert ist, auf dem Bildschirm mit den obigen 16 Textzeichen ausgibt. Achtung, die Wiederholungen können sich dabei über die Zeilenumbrüche des Bildes hinweg fortsetzen. Die Zeilenumbrüche sind in der Quelldatei nicht explizit vermerkt, sondern durch die Bildgröße eindeutig definiert.

Die Methode, Wiederholungen in einer Datei mit einem Zähler zusammenzufassen, heißt *Lauflängencodierung*. Sie wird sehr oft als erster Schritt in Kompressionsverfahren angewendet. Dabei kommt manchmal noch eine etwas raffiniertere Variante zum Einsatz, so dass einfache Vorkommen nicht mit einem Zähler versehen werden müssen.

Zusatzaufgabe (freiwillig): Wer ist auf den Bildern dargestellt? Tipp: Weit vom Bildschirm weggehen.

Aufgabe 16: Polynom-Interpolation **Pflichtaufgabe**

Bei numerischen Berechnungen tritt gelegentlich der Wunsch nach Funktionswerten von Funktionen auf, von denen nur einige Punkte bekannt sind. Dann müssen, mittels einer Modellannahme über die Funktion, die neu zu berechnenden Werte inter- oder extrapoliert werden. Ein möglicher Weg besteht darin, für n gegebene Punkte $(x_i,\, f(x_i))$, $i = 1, \ldots, n$ ein Polynom vom Grad $n-1$ zu bestimmen, und dieses Polynom für Vorhersagen zu verwenden.
Dies kann z.B. mittels der Interpolationsformel von Lagrange geschehen. Die Polyonome

$$L_i(x) := \prod_{\substack{k=1 \\ k\neq i}}^{n} \frac{x - x_k}{x_i - x_k}, \qquad i = 1, \ldots, n$$

sind alle vom Grad $n-1$ und haben die Eigenschaft

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 \text{ für } i = j \\ 0 \text{ für } i \neq j \end{cases}.$$

Damit ergibt sich für das Interpolationspolynom der Ausdruck

$$f(x) = \sum_{i=1}^{n} f(x_i) \cdot L_i(x) = \sum_{i=1}^{n} f(x_i) \prod_{\substack{k=1 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k}.$$

Die Funktion $f(x)$ ist vom Grad $n-1$ und nimmt an den Stellen x_i jeweils den Wert $f(x_i)$ an.

Scheiben Sie ein C++-Programm, das diesen Interpolationsausdruck für einzugebende Stützstellen x numerisch auswertet. In der Datei `/home/ck17/daten/a16-interpol.dat` oder auf der WWW-Seite finden Sie einen Datensatz. Verwenden Sie eine `while (datei >> ...)` Schleife, um aus der Datei Wertepaare $(x_i, f(x_i))$ in zwei Felder einzulesen, bis das Ende der Datei erreicht ist. Der Benutzer soll nun die Möglichkeit haben, x-Werte einzugeben, für die der interpolierte Funktionswert berechnet wird. Dies soll so lange geschehen, bis der Benutzer den Wert $x = 0$ eingibt. Geben Sie dabei jeden interpolierten Wert auf dem Bildschirm aus und schreiben Sie die Werte x und $f(x)$ zeilenweise in eine Ergebnisdatei `a16-interpol-res.dat`.
Die ursprüngliche Punktedatei und die von Ihnen erzeugte Ergebnisdatei schauen Sie sich bitte in einem geeigneten Plot-Werkzeug (z.B. `python/matplotlib`, `gnuplot` oder `xmgrace`) an. Stellen Sie bitte die Ausgabe so ein, dass die gegebenen und die neu berechneten Funktionswerte durch Punkte und verschiedene Farben erkennbar sind.

Zum Knobeln: Nim-Spiel **freiwillig**

Beim Nim-Spiel in der hier betrachteten Variante gibt es zwei Spieler und zwei Häufen mit Streichhölzern auf dem Tisch. Die Spieler sind abwechselnd am Zug und haben dabei jeweils folgende zwei Möglichkeiten: entweder von genau einem Haufen ihrer Wahl eine beliebige Zahl (≥ 1) an Hölzchen zu entfernen oder aber von beiden Häufen genau die gleiche Zahl an Hölzchen zu entfernen. In jedem Zug muss also immer mindestens ein Streichholz weggenommen werden. Wer mit seinem Zug das letzte Streichholz (oder die letzten Streichhölzer) wegnehmen kann, hat gewonnen.

Schreiben Sie zunächst ein C++-Programm, welches Sie dieses Spiel mit zwei Spielern spielen lässt. In diesem Fall soll das Programm also die Streichholzhäufen verwalten und dabei von beiden Spielern abwechselnd gültige Züge erfragen, bis ein Spieler gewonnen hat.

Interessant ist es, sich zu überlegen, welche Strategie in diesem Spiel zum Gewinn führt. Gibt es Haufengrößen, von denen aus der Sieg erzwungen werden kann?
Ersetzen Sie in Ihrem Programm einen Mitspieler durch den Computer und bringen Sie ihm Ihre Gewinnstrategie bei.

Hinweis: Es ist zweckmäßig, mit zwei zufälligen Stapeln von je etwa 5 bis 25 Streichhölzern zu beginnen. Um dies zu erreichen, können Sie vom Rechner ermittelte Zufallszahlen verwenden. Nach der Anweisung `#include <cstdlib>` und `#include <time.h>` im Kopf des Programms können Sie mit `rand()%a` ganzzahlige Zufallszahlen im Intervall $[0, a[$ erzeugen. Um den Generator auf einen nicht vorherbestimmten Anfangswert zu setzen, benötigen Sie noch ein einmaliges `srand((unsigned int)time(0))` am Anfang des Programmcodes.
