Dirk Praetorius
Christoph Überhuber

Wintersemester 2007/08
14. November 2007

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

Die Aufgaben mit Stern (*) sind bis zur nächsten Übung vorzubereiten und werden dort abgeprüft. Die übrigen Aufgaben dienen nur Ihrer Übung und mir als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code in ein Unterverzeichnis `serie06` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können.*

Aufgabe 51*. Schreiben Sie eine Funktion `float2dec`, die für eine gegebene Mantissenlänge $M \in \mathbb{N}$, Ziffern $a_1, \ldots, a_M \in \{0, 1\}$ und einen Exponenten $e \in \mathbb{Z}$ den Dezimalwert $x = \left(\sum_{k=1}^{M} a_k 2^{-k}\right) 2^e$ berechnet und zurückgibt. Schreiben Sie ein aufrufendes Hauptprogramm, in dem M, a_j und e eingelesen und der Wert x ausgegeben wird. Realisieren die Potenzen 2^{-k} möglichst rechenökonomisch. Speichern Sie den Source-Code unter `float2dec` ins Verzeichnis `serie06`.

Aufgabe 52*. Schreiben Sie eine Funktion `dec2float`, die für eine gegebene Dezimalzahl $x \in \mathbb{R}_{>0}$ und eine Mantissenlänge $M \in \mathbb{N}$ die Ziffern $a_1, \ldots, a_M \in \{0, 1\}$ und den Exponenten $e \in \mathbb{Z}$ der normalisierten Gleitkommadarstellung (d.h. $a_1 = 1$) berechnet und zurückgibt. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x eingelesen und die Gleitkommadarstellung von x ausgegeben wird. Um Ihre Funktion zu verifizieren, können Sie Aufgabe 51 verwenden. Speichern Sie den Source-Code unter `dec2float` ins Verzeichnis `serie06`.

Aufgabe 53*. Man schreibe eine Prozedur `pascal(k)`, die die ersten k-Stufen des Pascal'schen Dreiecks ausgibt: Jede Zeile dieses Schemas beginnt und endet mit 1. Die restlichen Zahlen werden als Summe nebeneinanderstehender Zahlen der vorhergegangenen Zeile gebildet. Für $k = 5$ gilt beispielsweise

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

Realisieren Sie das Pascalsche Dreieck möglichst rechenökonomisch, insbesondere ohne die Darstellung der Einträge über den Binomialkoeffizienten. Dies ist mit Hilfe *eines* Vektors $x \in \mathbb{N}^k$ der Länge k möglich. Schreiben Sie ferner ein Hauptprogramm, das $k \in \mathbb{N}$ einliest und die Prozedur aufruft. Speichern Sie den Source-Code unter `pascal` ins Verzeichnis `serie06`.

Aufgabe 54*. Schreiben Sie eine Funktion `unique`, die einen Vektor $x \in \mathbb{R}^n$ aufsteigend sortiert, doppelte Einträge streicht und den Vektor in gekürzter Form zurückgibt. Die Funktion soll also beispielsweise den Vektor $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ durch den Vektor $x = (1, 3, 4, 5) \in \mathbb{R}^4$ überschreiben. Die Länge n der Vektoren ist dynamisch zu realisieren. Schreiben Sie ein aufrufendes Hauptprogramm, in dem n und $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird. Speichern Sie den Source-Code unter `unique` ins Verzeichnis `serie06`.

Aufgabe 55. Gegeben sei ein Vektor $x \in \mathbb{R}^n$ und eine Schranke $k \in \mathbb{N}$. Man schreibe eine Funktion `cut`, die aus x alle Glieder x_j mit $|x_j| > k$ streicht und den gekürzten Vektor zurückgibt. Dies kann auf verschiedene Arten erfolgen:

- ohne Zusatzspeicher: Man geht den Vektor x von vorne durch, streicht die entsprechenden Glieder und kopiert die nachfolgenden Glieder um eine Stelle nach vorne. Am Ende muss x geeignet gekürzt werden.
- ohne Zusatzspeicher: Man geht den Vektor x von hinten durch, streicht die entsprechenden Glieder und kopiert die nachfolgenden Glieder um eine Stelle nach vorne. Am Ende muss x geeignet gekürzt werden.
- mit Zusatzspeicher: Man legt Zusatzspeicher für y in der Länge von x an und kopiert $x(j)$ nach y, falls $|x(j)| \leq k$ gilt. Am Ende muss y geeignet gekürzt und x gelöscht werden.
- mit Zusatzspeicher: Man geht zunächst x durch und zählt die Anzahl der Glieder $x(j)$ mit $|x(j)| \leq k$. Man legt dann y mit geeigneter Länge an und kopiert $x(j)$ nach y, falls $|x(j)| \leq k$ gilt. Am Ende muss x gelöscht werden.

Wie wird sich die Laufzeit dieser Varianten verhalten? Welche ist wohl die schnellste, welche die langsamste (zumindest wenn x vergleichsweise lang ist)?

Aufgabe 56. Das Δ^2-Verfahren von Aitkin ist ein Verfahren zur Konvergenzbeschleunigung von Folgen. Für eine injektive Folge $(x_n)_{n\in\mathbb{N}}$ mit $\lim_n x_n = x$ definiert man

$$y_n := x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}$$

Unter gewissen zusätzlichen Voraussetzungen an die Folge $(x_n)_{n\in\mathbb{N}}$ gilt dann

$$\lim_{n\to\infty} \frac{y_n - x}{x_n - x} = 0,$$

d.h. die Folge $(y_n)_{n\in\mathbb{N}}$ konvergiert schneller gegen x als $(x_n)_{n\in\mathbb{N}}$. Man schreibe eine Funktion `aitkin`, die für einen Vektor $x \in \mathbb{R}^n$ mit Länge $n \geq 3$ den Vektor $y \in \mathbb{R}^{n-2}$ berechnet.

Aufgabe 57. Um numerisch Nullstellen von Funktionen zu finden, formuliert man Probleme häufig in Fixpunktform: Die positive Nullstelle x^* der Funktion $f(x) = x^2 + \exp(x) - 2$ ist beispielsweise Fixpunkt der Funktion $\phi(x) = \log(2 - x^2)$, d.h. $\phi(x^*) = x^*$. Man definiert nun die Iteriertenfolge $x_n := \phi(x_{n-1})$ für einen Startwert x_0. Falls die Folge $(x_n)_{n\in\mathbb{N}}$ konvergiert, konvergiert sie aus Stetigkeitsgründen gegen einen Fixpunkt von ϕ und damit gegen eine Nullstelle x^* von f. Realisieren Sie diese Fixpunktiteration mit/ohne Aitkinsche Konvergenzbeschleunigung. Die Iteration soll abgebrochen werden, sobald $|f(y_n)| \leq \tau$ und $|y_n - y_{n-1}| \leq \tau \max\{|y_n|, |y_{n-1}|\}$ gilt, dabei bezeichnen y_n die Folgenglieder der Aitkin-Folge (bzw. $y_n = x_n$). Was viele Iterationsschritte beobachtet man in beiden Fällen?

Aufgabe 58. Für eine konvergente Folge $(x_n)_{n\in\mathbb{N}}$ mit Grenzwert x spricht man von Konvergenzordnung $p \geq 1$, falls es eine Konstante $c > 0$ gibt mit $|x_n - x| \leq c\,|x_{n-1} - x|^p$ für alle $n \in \mathbb{N}$. Mit dem Ansatz

$$|x_n - x| = c|x_{n-1} - x|^p \quad \text{und} \quad |x_{n-1} - x| = c|x_{n-2} - x|^p \quad \text{für } n \geq 2$$

kann man für fixiertes n die Unbekannten p und c bestimmen. Elementare Rechnung zeigt dann

$$p = \frac{\log(|x_n - x|/|x_{n-1} - x|)}{\log(|x_{n-1} - x|/|x_{n-2} - x|)} \quad \text{und} \quad c = \frac{|x_n - x|}{|x_{n-1} - x|^p},$$

d.h. aus den Gleichungen für die Fehler $|x_{n-1} - x|$ und $|x_n - x|$ berechnet man die Unbekannten p und c. Leiten Sie diese Gleichheiten her! Schreiben Sie eine Funktion `convorder`, die für eine gegebene Folge $(x_n)_{n=1}^N$ und einen Grenzwert x die experimentelle Konvergenzrate $p, c \in \mathbb{R}^{N-2}$ berechnet und zurückgibt. — Üblicherweise ist x unbekannt und man hat nur eine Folge $(x_n)_{n=1}^N$ von Approximationen. In diesem Fall kann man die Funktion `convorder` mit der Teilfolge $(x_n)_{n=1}^{N-1}$ und $x := x_N$ verwenden.

Aufgabe 59. Schreiben Sie die Fixpunktiteration aus Aufgabe 57 so, dass die Folge $(x_n)_{n=1}^N$ der Ierierten zurückgegeben wird, berechnen Sie die zugehörige Experimentelle Konvergenzordnung p sowie die zugehörige Konstante c laut Aufgabe 58, und geben Sie diese aus. Vergleichen Sie die numerischen Resultate mit anderen Iterationen, z.B. dem Bisektionsverfahren aus Aufgabe 33, dem Sekantenverfahren aus Aufgabe 82 und dem Newton-Verfahren aus Aufgabe 83.

Aufgabe 60. Wir betrachten ein Gleitkommazahlsystem $\mathbb{F}(2, M, e_{\min}, e_{\max})$ mit Mantissenlänge $M \in \mathbb{N}$ und Exponentialschranken $e_{\min}, e_{\max} \in \mathbb{Z}$. Was ist die größte normalisierte Gleitkommazahl $x > 0$? Was ist die kleinste normalisierte Gleitkommazahl $x > 0$? Wie viele normalisierte Gleitkommazahlen gibt es insgesamt, d.h. wie viele Zahlen enthält $\mathbb{F}(2, M, e_{\min}, e_{\max})$?