Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
18.11.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Aufgabe 8.1. Gegeben seien die Summen

$$a_N := \sum_{n=0}^{N} \frac{1}{(n+1)^2} \quad \text{und} \quad b_M := a_M^2 = \sum_{m=0}^{M} \sum_{k=0}^{m} \frac{1}{(k+1)^2(m-k+1)^2}.$$

Schreiben Sei ein Programm, welches für verschiedene Werte von N bzw. M die Zeit misst um a_N bzw. b_M zu berechnen. Geben Sie anschließend die Ergebnisse in Form einer Tabelle am Bildschirm aus. Entsprechen die Resultate Ihren Erwartungen? Speichern Sie den Source-Code unter `zeitmessung.c` in das Verzeichnis `serie08`. *Hinweis:* Überlegen Sie sich wie groß der Aufwand bei der Berechnung von a_N bzw. b_M ist. Wählen Sie zunächst kleine Werte von N und M.

Aufgabe 8.2. Um aus experimentellen Daten das asymptotische Verhalten zu bestimmen, kann man wie folgt vorgehen. Gegeben sei eine Funktion $f : \mathbb{N} \to \mathbb{R}$ und Datenpunkte $(N_i, f(N_i))$ für $i = 0, \dots M$. Angenommen $f = CN^\alpha$ mit einer Konstante $C > 0$ und $\alpha \in \mathbb{R}$. Aus den Datenpunkten lassen sich dann durch

$$\alpha_i = \frac{\log(f(N_i)) - \log(f(N_{i-1}))}{\log(N_i) - \log(N_{i-1})} \quad \text{für } i = 1, \dots, M$$

experimentelle Raten bestimmen. Überlegen Sie sich wie man auf obige Formel kommt. Schreiben Sie ein Programm, welches aus den Daten der vorigen Aufgabe Raten für die Berechnung der beiden Summen bestimmt. Der Wert $f(N)$ ist also die Zeit zur Berechnung der Summe a_N. Analoges gilt für die zweite Summe. Geben Sie Ihre Ergebnisse in Form einer Tabelle aus. Stimmen die Raten mit Ihren Erwartungen überein? Speichern Sie den Source-Code unter `raten.c` in das Verzeichnis `serie08`.

Aufgabe 8.3. 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

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

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

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

d.h. aus den Gleichungen für die Fehler $|x_{n+2} - x|$ und $|x_{n+1} - x|$ berechnet man die Unbekannten $p_n := p$ und $c_n := c$. 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. Speichern Sie den Source-Code unter `convorder.c` in das Verzeichnis `serie08`.

Aufgabe 8.4. Eine Variante zur Berechnung einer Nullstelle einer Funktion $f : [a, b] \to \mathbb{R}$ ist das *Newton-Verfahren*. Ausgehend von einem Startwert x_0 definiert man induktiv eine Folge $(x_n)_{n\in\mathbb{N}}$ durch

$$x_{k+1} = x_k - f(x_k)/f'(x_k).$$

Man realisiere das Newton-Verfahren in einer Funktion `newton`, wobei die Iteration abgebrochen wird, falls entweder

$$|f'(x_n)| \leq \tau$$

oder

$$|f(x_n)| \leq \tau \quad \text{und} \quad |x_n - x_{n-1}| \leq \begin{cases} \tau & \text{für } |x_n| \leq \tau, \\ \tau|x_n| & \text{sonst} \end{cases}$$

gilt. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` und Ableitung `double fstrich(double x)` arbeiten. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 eingelesen und x_n ausgegeben wird.Speichern Sie den Source-Code unter `newton.c` in das Verzeichnis `serie08`.

Aufgabe 8.5. Testen Sie das Newtonverfahren aus der vorigen Aufgaben mit Polynomen und deren Ableitungen. Schreiben Sie dazu Funktionen, um ein Polynom bzw. dessen Ableitung an einem beliebigen Punkt auszuwerten. Wie Sie Polynome mit Hilfe von Arrays darstellen können, haben Sie bereits kennengelernt. Verwenden Sie auch Polynome mit mehreren Nullstellen und variieren Sie die Startwerte. Was passiert bei dem Polynom $(x-1)^2 - 1$ und einem Startwert nahe bei $x = 1$? Speichern Sie den Source-Code unter `testNewton.c` in das Verzeichnis `serie08`.

Aufgabe 8.6. Für eine differenzierbare Funktion $f : [a, b] \to \mathbb{R}$ kann man die Ableitung $f'(x)$ in einem festen Punkt $x \in \mathbb{R}$ durch den einseitigen Differenzenquotienten

$$\Phi(h) := \frac{f(x+h) - f(x)}{h} \quad \text{für } h > 0$$

approximieren. Schreiben Sie eine Funktion `double* diff(double x, double h0, double tau, int* n)`, die für $h_n := 2^{-n} h_0$ die Folge der $\Phi(h_n)$ berechnet, bis gilt

$$|\Phi(h_n) - \Phi(h_{n+1})| \leq \begin{cases} \tau & \text{falls } |\Phi(h_n)| \leq \tau, \textit{oder} \\ \tau\,|\Phi(h_n)| & \text{anderenfalls.} \end{cases}$$

Die Funktion liefere in diesem Fall die vollständige Folge $(\Phi(h_0), \ldots, \Phi(h_n))$ der Iterierten zurück. Beachten Sie, dass auch die Länge des Vektors „zurückgegeben“ werden muss. Speichern Sie den Source-Code unter `diff.c` in das Verzeichnis `serie08`.

Aufgabe 8.7. Genauso wie der Inhalt von Variablen elementaren Datentyps kann auch der Inhalt eines Pointers mittels `printf` ausgegeben werden. Man verwendet hier `%p` als Platzhalter für Adressen. Die Ausgabe dafür erfolgt systemabhängig meist in Hexadezimaldarstellung. Schreiben Sie eine Funktion `void charPointerAbstand(char* anfangsadresse, char* endadresse)`, welche folgende drei Werte tabelliert:

- Anfangsadresse
- Endadresse
- Abstand (Differenz) der beiden Adressen (Platzhalter im `printf` beachten!)

Da Arrays zusammenhängend im Speicher liegen, entspricht der Abstand zweier aufeinanderfolgender Elemente genau dem Speicherverbrauch des entsprechenden Datentyps. Testen Sie Ihre Funktion für einen `char`-Array `c[2]` mit den beiden Aufrufen:

```
charPointerAbstand(&c[0],&c[1]);
charPointerAbstand(c,c+1);
```

Schreiben Sie nun nach obiger Manier eine Funktion `void doublePointerAbstand(double* anfangsadresse, double* endadresse)`, testen diese mit einem `double`-Array und vergleichen die unterschiedlichen Ergebnisse. Was passiert, wenn `charPointerAbstand` mit Adressen von `double`-Variablen aufgerufen wird? Speichern Sie den Source-Code unter `datentypen.c` in das Verzeichnis `serie08`.
Bonus: Finden Sie heraus, wieviel Speicher die Typen `short`, `int` und `long` auf dem Übungsserver verbrauchen.

Aufgabe 8.8. Welche Arten von Kommentaren gibt es? Was gibt folgender Code aus und warum?

```
#include <stdio.h>

/*int f(double x) {
    return (int) x;
  }
*/

main() {
  int x = 4;
  int y = 2*x*/* f(0.1)+3
              */1/4;
  // y = 1;
  printf("y = %d\n",y); // Ausgabe
}
```